Methods in Percolation

A thesis submitted in partial fulfilment
of the requirements for the Degree of

Doctor of Philosophy
in
Mathematical Physics

in the
University of Canterbury

by

Michael James Lee

University of Canterbury
2008

Abstract

Algorithms are presented for the computationally efficient manipulation of graphs. These are subsequently used as the basis of a Monte Carlo method for sampling from the microcanonical ensemble of lattice configurations of a percolation model within a neighbourhood of the critical point.

This new method arbitrarily increments and decrements the number of occupied lattice sites, and is shown to be a generalisation of several earlier, purely incremental, methods. As demonstrations of capability, the method was used to construct a phase diagram for exciton transport on a disordered surface, and to study finite size effects upon the incipient spanning cluster.

Application of the method to the classical site percolation model on the two-dimensional square lattice resulted in an exceptionally precise estimate of the critical threshold. Although this estimate is not in agreement with earlier results, its accuracy was established through an application specific test of randomness, which is also introduced here. The same test suggests that many earlier results have been systematically biased due to the use of deficient pseudorandom number generators. The estimate made here has since been independently confirmed.

Acknowledgements

I would especially like to thank Bob Ziff for numerous helpful discussions and suggestions regarding percolation theory, pseudorandom number generators, open problems, and data analysis. He was also kind enough to provide a pre-publication copy of his algorithms chapter from the Encylcopedia of Percolation [293], which has been an invaluable resource throughout the preparation of this manuscript.

I would also like to thank Youjin Deng, Henk Blöte, João Plascak, Paulo Martins, Chin-Kun Hu, Richard Brent, Robert Parviainen, and Paulo Oliveira for providing details of their respective pseudorandom number generators, Tadao Takaoka for discussions of search algorithms, Bill Taylor for discussions of randomness, Mike Reid, David Wojtas, Ewan Orr, and Richard Graham for general idea bouncing, Orlon Petterson, Dave van Leeuwen, Colin McMurtrie, and Tony Dale for computer access, Jennifer Gannon and Bill Broadley for their introductions to C and MPI, Kate Monahan for her tricks with LaTeX, Dharamvir Ahluwalia for insight into the publishing process, Scott Davidson for handling the bureaucracy, Bill Moreau for the OE, Walter Wyss for the potato method, Jim McConnell for the spirit of the Worm Runner's Digest, and Lennard Zinn for the wild goose chase.

Contents

Chapter 1

Introduction

The original research objective herein was to construct a dynamical cellular automaton model for the two-dimensional transport of energy excitons across the surface of an inert substrate doped with randomly scattered sites, each capable of hosting a single exciton [74] (see section 4.1). The model would predict the rate and direction of transport under various conditions, and so could be tested against observations from photon echo experiments [215]. Several models were developed, and although these produce reasonable qualitative behaviour, they invariably encountered conceptual difficulties once multiple excitons were present in the system. Since the laser induced excitations intrinsic to the experiments necessarily involve high exciton densities, these cellular automaton models were of dubious physical accuracy and could be certain to make incorrect predictions for those same experimental tests. As is detailed in section 4.1.2, it turns out that this is a well documented problem and the essential flaw is the concept of the exciton itself; nearby excitons are strongly interacting, do not behave as individual quasiparticles, and collective motion is important [30, 103, 128]. Consequently, any discrete exciton approach to dynamical modeling, automaton or not, is bound to fail whenever multiple adjacent host sites are simultaneously activated.

Provided that less information is sought, some answers can be obtained by studying the underlying transport medium rather than the excitons themselves. This avoids the problem of exciton physics altogether, and although dynamical quantities such as the precise exciton drift velocity cannot be calculated, it remains possible to determine the transport phase of

the system (that is, whether or not long range transport actually occurs) [5]. It turns out that this is a well studied quantum percolation exercise [33, 135, 183, 194, 233, 235]. In the process of constructing a method to determine these transport phase diagrams, it became apparent that the concept behind Tomita and Okabe's probability-changing cluster algorithm for the Potts model [263] could be adapted for use with the classical and quantum percolation models. The key to making this work was finding an efficient mechanism for unoccupying sites and bonds on a lattice (earlier techniques have been concerned exclusively with the occupation of these entities).

The final product, described in chapter 2, was a pair of computationally efficient algorithms for the merging and splitting of graphs by the insertion and removal of edges between vertices. These algorithms employ a data structure in which graphs are represented by sets of connected vertices (general graph structures) that are augmented with overlying tree structures. These trees provide an efficient mechanism for uniquely identifying each graph, and the introduction of multiple trees provides a rapid means for isolating and identifying the fragments resulting from the removal of one or more graph edges. Clearly, the ability to add and remove edges and vertices permits the arbitrary modification of any graph. While this has many potential applications, the one explored here was to generalise the unidirectional (vertex addition) classical percolation model microcanonical ensemble lattice configuration Monte Carlo sampling method to a bidirectional (addition and subtraction) method [152] (the data structures used by the new algorithms are essentially a generalisation of those employed by Newman and Ziff [190, 191] in their efficient implementation of the unidirectional approach [73, 72, 87, 164, 163]).

Chapter 3 forms the core of this thesis, in which the algorithms are applied to the problem of determining the site percolation threshold of the two dimensional square lattice [246]. At the outset, this was intended to be merely a short side venture, made only to demonstrate the new method's capabilities. However the problem, simple and elegant on the surface, is ridden with hidden subtleties and practical difficulties [22, 293]. Whether viewed from a mathematical, statistical or computational perspective, that is a seductive combination indeed.

Initial calculations were constrained by a severe lack of computing power

and data storage capacity. Furthermore, as data came back from the machines, the combined threshold estimate appeared to drift very slowly downward. This was somewhat perplexing of course, but the problem was eventually traced to a disagreement between the (individually stable) estimates resulting from each of the two supposedly adequate pseudorandom number generators that were being used (and the fact that one was returning more data). Eventually two mutually incompatible estimates for the threshold were produced and, hesitantly, published [152]. The first of these was based upon the same generator that had been used by Newman and Ziff to calculate the then best accepted value [190, 191]. Although it lay above the Newman-Ziff result, this first estimate was compatible with that value (although of lower precision) and, in any case, the Newman-Ziff result was believed to be a little too low itself (on the grounds of other contemporary estimates [46, 196, 296]). The trouble was that the second new estimate, although of higher precision than the Newman-Ziff result, lay well below it, and well outside of the region within which the true value was believed to lie (see table 3.3).

The reaction to the published estimates was that the more precise value had to be too low. It was suggested that the two incompatible estimates should be combined, thereby producing something within the expected range. Of course the possibility of an error existed, but these were methodically ruled out, one by one, during the ensuing months. The suggestion was then raised that the generator used for the second estimate might be at fault. The trouble with this was that the generator upon which the second estimate was based was modern, of good quality, well tested, and highly regarded [177, 149]. In comparison, the generator used by Newman and Ziff, although good, was of a much older class that had established deficiencies [149, 176, 291]. Even so, belief in the expected region was strong enough that shortcomings of the newer generator were raised as a serious possibility.

Shortly thereafter, access was granted to the university's new Blue Gene computer facility, so providing a ten-fold increase in the number of processors available. This allowed for several different generators to be tested against each other. However, if threshold estimates from two different generators were found to be incompatible, it would not be obvious as to which of the generators (or both) was inadequate for the calculation. Hence it was suggested that results should be obtained from as many generators (and by

as many methods) as possible, and then averaged. This is fine if generator induced biases are symmetrically distributed either side of zero, but that seems optimistic. Nevertheless, the authors of contemporary threshold estimates were contacted and they were all kind enough to provide the details of their generators (this information is typically absent from the literature, refer to table 3.3). As it turned out, all the high precision Monte Carlo estimates since 2000 had been based upon generators of the class used by Newman and Ziff (or very close to). This meant that the expected region for the percolation threshold was being unduly weighted by several estimates that might well all have the same flaw, and hence there need not be anything wrong with the estimates obtained here (from the new algorithms and the modern generator) after all. Subsequent calculations consistently reproduced (to within statistical uncertainties) previously published threshold estimates (and the discrepancies between them) from each of the generators [153].

A method was then devised by which the deficient generators could be clearly identified [153]. In this test, the same calculation is performed at least twice under different enumerations (that is, different numerical indexings) of the model lattice. Since generator induced bias is the result of interference between numerical patterns in the generator output sequence and spatial patterns in the lattice enumeration, if a generator suffers from significant bias then the results obtained from it will change in response to the lattice enumeration. This is a computationally intensive process of course, but it provides an application specific reassurance of accuracy where previously there was none. Such tests, particularly of multiple generators at high precision, are not really feasible without massively parallel computer facilities, but such devices are now becoming increasingly common.

In the end, the suspicious estimate from the first set of calculations was completely validated, and the evidence suggests that earlier results have indeed suffered from generator induced systematic errors. All told, it took two years of head scratching, twelve hundred processors, and over a quadrillion pseudorandom numbers, but the final outcome of this meticulous numerical experiment was the first measurement of an unknown percolation threshold to achieve seven significant figures of precision [153]. The accuracy and reliability of the underlying Monte Carlo data has been ensured through the above testing. Also, by obtaining accurate high precision data from large

lattices, typical problems of scaling exponents and higher order corrections were completely avoided (there is a tendency within the literature to ignore all sources of uncertainty other than statistical fluctuation, and to quote this as the final uncertainty, when in practice it is merely a lower bound). Consequently the result ought to be completely reliable, and independent confirmation of this came only a week after publication [66]. The "wrong" threshold is now established. The main point, however, is that even today, Monte Carlo results may be incorrect or misleading because of inadequate pseudorandom number generators, and a means now exists for eliminating this problem.

In chapter 4, the manuscript returns to the original transport problem and calculates the site-bond percolation transport phase diagram for a substitutionally disordered medium (chronologically, this work was done in-between publication of the first set of estimates, from section 3.5, and the arrival of the Blue Gene facility in section 3.6). This was a simple task with the algorithms of chapter 2. Studies of boundary effects upon the incipient spanning cluster were then begun (these are important in cellular automata and bootstrap percolation [94, 226] where boundary effects remain significant on even enormous lattices [96]), but did not proceed very far before the latter stages of chapter 3 took over.

Chapter 2

Algorithms for Graphs

A pair of complementary algorithms are presented. One of the pair is a fast method for connecting graphs with an edge. The other is a fast method for removing edges from a graph. Both algorithms employ the same representation for graphs, and so, in concert, can arbitrarily modify any graph.

2.1 Graphs

A graph is a mathematical structure composed of a nonempty set of points, known as vertices or nodes, and a (possibly empty) set of line segments, known as edges, connecting pairs of those points [273]. The number of vertices in a graph is called its order. All possible graphs of order less than four are shown in figure 2.1. Graphs are of fundamental importance in computer science, where any given data structure can be represented as a graph or directed graph, and so the problem of locating specific data elements or pathways between them reduces to search algorithms on graphs [43]. Indeed. computer networks may themselves be represented by graphs [27, 189]. In physics, graphs appear as Feynman diagrams, and any discretised spatial structure can be represented as a graph. Hence the ubiquitous lattices of high energy physics and materials engineering reduce to graphs. There are numerous other applications of course, and many of these will be mentioned in chapter 3. The remainder of this chapter will be used to introduce a pair of complementary algorithms for the manipulation of graphs, without considering any specific application in detail.

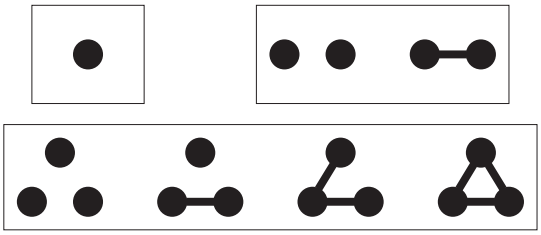

Figure 2.1: All graphs, connected and diconnected, of orders one, two and three (after Weisstein [273]). The number of possible different graphs of a given order increases rapidly with order.

2.2 Fundamental Data Types

The algorithms employ three fundamental data types; vertex objects, graph objects and clump objects. Note that it would be possible to introduce a separate class of edge objects if so desired, or even to do away with the vertex objects entirely and work purely with edge objects instead. Because the final implementation (as per the application of chapter 3) was written with vertex objects as fundamental and edges as an attribute of those objects, the following description shall reflect that choice.

2.2.1 Vertex Objects

Vertex objects represent vertices. Each vertex object will carry a number of internal data elements. These will generally include a set of pointers to other vertex objects (representing graph edges), a graph object pointer (to indicate graph structure membership), a clump object pointer (to indicate clump structure membership), and an additional pair of vertex object pointers to locate other vertex objects within the same clump structure. Other internal elements can be included as necessary, for example a set of spatial coordinates if geometry is important.

2.2.2 Graph Objects

Graph objects serve as a dynamic Hoshen-Kopelman relabelling table [115] for the vertex objects and form the backbone of graph structures. They are not, strictly speaking, required but allow for the tracking of greater quantities of data pertaining to each graph without consuming excessive amounts

of computer memory. They also simplify the edge removal algorithm and improve its performance. A graph object must include a pointer to other graph objects (for the merging of disconnected graphs). If the application calls for the set of graph objects to be periodically searched then typically another pair of graph pointers will be required to maintain them in a doubly-linked list or similar collection. Graph objects may include additional data elements for tracking observational quantities such as the order and geometrical extent of the graph.

2.2.3 Clump Objects

Clump objects serve essentially the same purpose as graph objects, but are employed by the edge removal algorithm to represent partially formed graphs. Whereas a graph structure always contains every vertex that is a member of that graph, a clump structure does not. As with graph objects, clump objects must contain a pointer to other clump objects for the merging of disconnected clumps. Additionally, clump objects must also contain an integer holding the order of the clump (also for the merging of disconnected clumps), and four vertex object pointers (for tracking internal and perimeter vertices of the clump). Other than order, clump objects need not contain the observational quantities data found on graph objects. Clump objects are short lived entities that exist only during the search phase of the edge removal algorithm (subsection 2.4.4).

2.3 Compound Data Types

The three fundamental data types can be combined into two compound data types; graph structures and clump structures. The tree structures below are not unique representations, but have been chosen to optimise performance of the algorithms in section 2.4.

2.3.1 Graph Structures

Connected graphs are represented by trees of vertex objects and graph objects (see figure 2.2). Disconnected graphs are simply collections of connected graphs. Each vertex object contains a pointer to the graph object of which it is a child (since every vertex must belong to some graph). Each

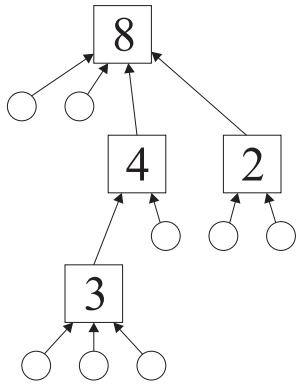

Figure 2.2: Possible data structure for a connected graph of order eight. Arrows indicate pointers between objects. Each graph object, □, has a counter for the number of vertex objects, ◯, in the subtree beneath it. Edges between vertex objects are not shown, and so there is no indication as to exactly which order eight connected graph is being represented.

graph object may contain a pointer to some other graph object of which it is a child (whenever some graph is a substructure of some larger graph). Trees are constructed in such a way that for each tree there always exists one, and only one, graph object which is not the child of any other object and so has no graph object pointer (or, more typically, has a null pointer). This unique object is designated as the root of the tree. Graph objects form the backbone of the structure while all leaves of the tree are vertex objects and all vertex objects are leaves. All vertex objects within a given tree are members of the same graph. It is not true, other than for the root, that the set of all vertex objects belonging to the sub-tree below a given graph object correspond to the vertices of a connected subgraph (because of the path compression applied by the algorithm of subsection 2.4.1).

2.3.2 Clump Structures

During the accretion process of the edge removal algorithm of subsection 2.4.4, partially reconstructed graphs are represented by trees of clump objects and graph objects. Clump structures are used so that the parent graph structure need not be tampered with. Although all vertex objects are, at all times, members of some graph structure, vertex objects only belong to clump structures on a temporary basis following the removal of an edge from

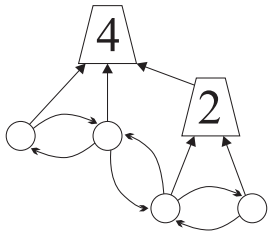

Figure 2.3: Possible data structure for a clump of four vertices. Each clump object (trapezoid) has a counter for the number of vertex objects (circles) beneath it. All vertex objects in the structure are kept within a doubly linked list. Edges between vertices are not shown.

the connected graph of which the vertices were members. A key difference between clump structures and graph structures is that all vertex objects belonging to a single clump structure are connected in a doubly-linked list. The significance of this will be discussed in subsection 2.4.4. Otherwise, as shown by figure 2.3, clump structures are essentially identical to graph structures.

2.4 Graph Manipulation Algorithms

Here a set of fast algorithms for the arbitrary manipulation of graphs is presented. It should be apparent that, as the product of a process of convergent evolution, the edge insertion algorithm of subsection 2.4.3 (using as it does the root find algorithm of subsection 2.4.1), is essentially a variation on the very efficient weighted union/find with path compression algorithm [111, 43] as used, for example, by Newman and Ziff [190, 191]. The edge removal algorithm of subsection 2.4.4 uses breadth-first searching [43] and weighted union/find with path compression to efficiently determine the daughter (fragment) graphs resulting from the removal of one or more edges from some (initial) parent graph. The use of graph and clump objects allows both of these algorithms to operate efficiently upon a common data structure, and allows easy adaptation to applications with differing observable quantities. The trade-off is that these structures are somewhat inefficient in terms of their computer memory requirements.

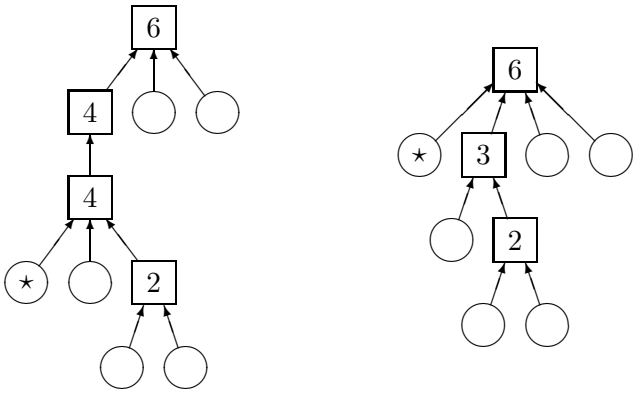

Figure 2.4: The left hand tree of graph objects □ and vertex objects ◯ represents a graph. Data upon each graph object tracks the total number of vertex objects below. From the marked ⋆ vertex object the root is found by a recursive algorithm which follows pointers to the root and turns every object along the way into a child of the root. The result is the right hand tree which is a simpler representation of the same graph. The graph object which was a child of the root in the left hand structure has been pruned from the tree.

2.4.1 Root Find Algorithm

The edge insertion and removal algorithms of subsections 2.4.3 and 2.4.4 work with vertex objects and need to know which graph and clump structures those objects belong to. Any graph or clump structure contains a single root object unique to that structure (see section 2.3), and so this serves as a means of unambiguously labelling the structure as a whole. Hence the problem of identifying a graph or clump is equivalent to finding the root object of the corresponding data structure (beginning with some given vertex object). For every vertex object within the structure, there exists a sequence of graph or clump objects ending with the root. For every object within the sequence, a pointer upon that object uniquely specifies the next object in the sequence (as per section 2.2). Starting with a vertex object, the sequence is found by tracing pointers from child to parent until the root is reached.

A path compression technique [111, 190, 191] is applied throughout the tracing process by redirecting the pointer of every object in the sequence (other than the root) to the location of the root (see figure 2.4). In this way all objects in the sequence are made children of the root and future

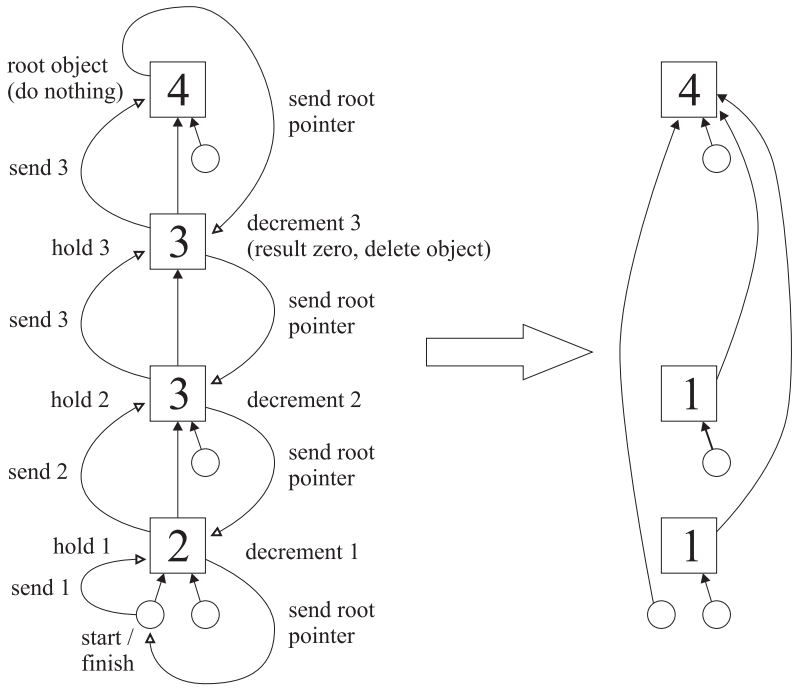

Figure 2.5: Details of a recursive implementation of the root find algorithm. Beginning with the indicated vertex object (◯), order data is transmitted up the tree (hollow headed arrows) by following a path of pointers (solid headed arrows) through the backbone of graph objects (□), and then the root object pointer is sent back down the same path. The end result is a simpler structure where every graph object knows how many vertex objects lie beneath it. Any graph objects with no children are deleted.

tracing operations from those objects will find the root after following only a single pointer. This affords a significant improvement in computational efficiency, where the average time to find the root of a tree of n objects goes to a constant as n approaches infinity [257, 190, 191]. To maintain correct data upon each graph or clump object, relevant information is transferred from child to parent on every step of the tracing. After path compression, some graph objects from the sequence may have no children. These redundant objects are pruned from the tree and deleted from memory during the tracing process. This is necessary for the algorithms to run indefinitely within finite memory, and eliminates any possibility of data substructures becoming detached (orphaned) from the whole following path compression. Details of a recursive implementation are shown in figure 2.5.

An equivalent iterative form is listed in script 2.1. The extent and nesting of conditional expressions is indicated, Python style, by indentation. Conditionals are tested only at the top of each loop. As noted, all objects have a counter that tracks the number of vertex objects on the sub-tree beneath (and including) themselves. The "and including" is significant as it supplies all vertex objects with an implied count of unity (other objects have an intrinsic count of zero). Hence this count can be referred to as a mass. Vertex objects have an intrinsic mass (order) of unity, other objects have an intrinsic mass (order) of zero. The mass on the root object is the order of the graph. Stacks are first in last out assemblies, objects are removed from the stack upon retrieval. In practice, one would normally stack pointers or tokens for objects rather than the objects themselves.

```
define rootof(object):
    % Follow pointers to root object
    place zero on mass-stack
    while object's parent exists
        place object on object-stack
        place object's mass on mass-stack
        substitute object's parent for object
    set root equal to object
    % Apply path compression, update mass
    % data, and delete redundant objects
    retrieve mass from mass-stack (discard)
    while mass-stack is non-empty
        retrieve mass from mass-stack
        retrieve object from object-stack
        set object's parent equal to root
        decrement object's mass by retrieved mass
        if object's mass is zero
            delete object
    return root
```

Script 2.1: Root finding algorithm. Locates the root of a tree from any starting point (leaf or node object), applies path compression to make future searches faster, maintains correct sub-tree mass data, and prunes redundant (graph or clump) objects. Vertex objects have a mass count of unity. The mass count on other objects is the sum of the masses of all the vertex objects on the sub-tree beneath them. Stacks are first in last out. It is possible to re-express this iterative implementation in a functionally equivalent, yet more compact, recursive form.

2.4.2 Vertex Extraction Algorithm

During the course of the edge removal algorithm (subsection 2.4.4), it will sometimes be necessary to remove vertex objects from graph structures since graph structures represent connected graphs and vertices may become disconnected from the parent graph as a result of removing an edge. In order to do this, the root find algorithm (subsection 2.4.1) is first applied so that the identity of the parent graph structure is known and the vertex to be removed is guaranteed to be a direct child of the root object. Having done this, the pointer from the vertex object to the root object is redirected toward some new graph object that is the root of a new graph structure into which the disconnected vertex is to be transferred (or else, if desired, the vertex object may be deleted). Data, such as graph order (see section 2.2), is then adjusted, from information upon the transferred vertex object, on both roots. More than one vertex object can be transferred at a time (see figure 2.6). It is the job of the edge removal algorithm to ensure, that when all vertex objects have been transferred, each of the resulting graph structures represents a connected graph, and all graph structures are disconnected from one another (otherwise the result would include at least one graph structure with two or more roots).

An implementation of this algorithm is specified in script 2.2. The form listed is more general than vertex extraction, and can prune any sub-tree. Naturally, since all vertex objects are leaves of the tree, a single vertex object constitutes a complete sub-tree. If the original tree is left containing no vertex objects after pruning of the sub-tree, then the `rootof` function (defined in script 2.1) ensures that what remains of the original tree is only a single (root) object (necessarily with a mass of zero). Any such massless trees are deleted, as the empty graph does not exist. Since, in this listing, the extracted sub-tree is not automatically made the child of some other object, a single extracted vertex will not immediately belong to any graph tree (even though the vertex exists and a single vertex is still a graph). However, since the vertex could be destined for deletion, assignment to a new (single vertex) graph, or transference to another (pre-existing) graph, it is more efficient to deal with this on a case by case basis outside of the extraction process. Furthermore, this implementation performs no checks to ensure that no edges exist connecting any of the extracted vertices to any of the remaining vertices (again, this is for efficiency).

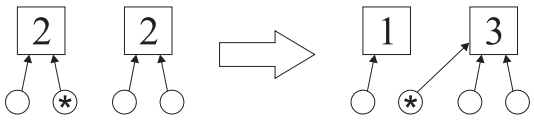

Figure 2.6: To extract a vertex object from a graph structure begins with the simple matter of locating the root object. The root find algorithm guarantees that the vertex object is now a direct child of the root object. The vertex object is removed from the structure by redirecting its pointer elsewhere, say to the root of another structure, and updating root counters accordingly. This figure shows the marked vertex object being moved from one struture to another.

```
define extract(object):
    % Ensure vertex is a direct child of the root object
    set root equal to rootof(object)
    % If the object is not itself the root, remove it from the tree
    if object's parent exists
        set object's parent to null
        decrement root's mass by object's mass
        % If the original tree contains no vertices, delete it
        if root's mass is zero
            delete root
    return
```

Script 2.2: Sub-tree extraction algorithm. The sub-tree beneath (and including) the specified object is pruned from its tree. If this results in a massless tree (empty graphs), that tree is deleted. The function `rootof` is defined in script 2.1. The operation "set object's parent to null" should be interpreted as "define this object to have no parent".

2.4.3 Edge Insertion Algorithm

The simplest possible graph consists of a single vertex and no edges (see section 2.1). Such a graph is represented here by a single vertex object with its pointer, of subsection 2.2.1, directed at a graph object that has no other children. More complex graphs arise when the vertices of these simplest graphs become connected by edges. Creating an edge is a simple matter of directing pointers from each of the associated pair of vertex objects to the other. The root find algorithm (subsection 2.4.1) is then applied to both of these vertex objects. If the objects belong to different graph structures then the corresponding disconnected graphs have been connected by the edge and the graph structures must be consolidated. This is achieved by directing a pointer from the root of the lower order graph to the root of the higher order graph, thereby making the lower order root a child of the higher order root and the lower order structure a sub-tree of the higher order structure (absorbing a smaller tree into a larger tree in this way is known as weighting [111, 190, 191, 43]). Statistics upon the one remaining root are incremented by those on the former root so that the data upon the remaining root correctly reflects the properties of the new graph. As shown in figure 2.7, the result is a single tree with a unique root.

The procedure listed in script 2.3 details the tree consolidation component of the edge insertion process, This is simply a weighted union algorithm, preceeded by a check to ensure that two disjoined trees have been specified. The actual insertion of an edge between two vertices is a trivial matter typically consisting of directing pointers from each vertex to the other.

2.4.4 Edge Removal Algorithm

An edge may be removed from between two vertices by deleting the appropriate pointers from the corresponding vertex objects. A vertex may be removed from a graph by removing all of the edges about that vertex (the associated vertex object may then be transferred to a new graph, or else deleted, by the vertex extraction algorithm of subsection 2.4.2). Such operations may cause the original graph to fragment into a set of smaller graphs. The deterministic accretion algorithm which efficiently identifies these fragments is similar to the stochastic methods of Hammersley [105], Leath [146] and Alexandrowicz [4].

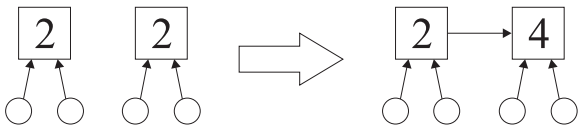

Figure 2.7: Whenever two vertex objects belonging to different graph structures become connected by an edge, the roots of the two structures are found and the root of the lesser order structure is made a child of the root of the greater order structure, whos graph order counter is incremented by that of the lesser structure. Edges are not shown in this diagram.

```
define consolidate(object1,object2):
    % Find root object (tree identifier) for each object
    set root1 equal to rootof(object1)
    set root2 equal to rootof(object2)
    % If the objects belong to different trees, make
    % the smaller tree a sub-tree of the larger tree
    if root1 does not equal root2
        if root1's mass exceeds root2's mass
            set root2's parent equal to root1
            increment root1's mass by root2's mass
        else
            set root1's parent equal to root2
            increment root2's mass by root1's mass
    return
```

Script 2.3: Tree consolidation algorithm. If the two objects are found to belong to two different trees, then the smaller tree is made a sub-tree of the larger (in order to minimise the search time in any future calls of `rootof` (defined in script 2.1)). After this function is called, the two objects are certain to belong to a common tree. Hence this procedure should normally be performed upon a pair of vertex objects immediately before or after inserting an edge between them.

Each surviving vertex that has had one or more of its edges removed is assigned to a distinct clump. A clump is merely a label held in common by a set of vertices which are all known to belong to the same fragment graph. Each such labelled vertex forms a separate nucleation kernel for an accretion process which constructs the fragment graphs from the remains of the original. From each labelled vertex, all edges are followed outward to what must be an adjacent vertex and these are assigned to the same clump as the labelled vertex. As vertices are added to the perimeter of a clump, they are placed in a queue to be examined for the presence of their own adjacent vertices (which will also need to be added to the clump).

This process is performed in parallel for all clumps, and is essentially a multiple simultaneous breadth-first pathway search through the graph [43]. In general, breadth-first searching is going to be the fastest method for finding a connected pathway between two vertices [256, 43]. Tests with the Monte Carlo sampling application of subsection 3.4.3 confirmed that depth-first searching is indeed much slower than breadth-first searching (by a factor of between two and three). However, if additional information is available regarding the structure of the graph, then more efficient weighted searches may become possible [43].

Clump labelling is achieved by the use of clump objects to which vertex objects may or may not have pointers, exactly as with graph structures. When two adjacent vertex objects are found to belong to different clump structures (by the algorithm of subsection 2.4.1), those two structures are merged into one. Merging clump structures is essentially the same process as connecting graph structures, but with small modifications to handle the perimeter object search queue. This is shown in figure 2.8.

When a clump contains no vertex that is adjacent to any other vertex that is not a member of the same clump, then that clump has stopped growing and must represent a complete connected fragment graph. All vertex objects associated with the clump are then extracted from the original graph structure (by the algorithm of subsection 2.4.2) and are made children of a newly created (root) graph object. Parameters for this fragment graph are calculated upon the root object and the clump ceases to be.

When the number of distinct clumps remaining drops to one, whatever remains of the original graph tree must logically represent the final fragment graph. It is not necessary to continue with the accretion process, what

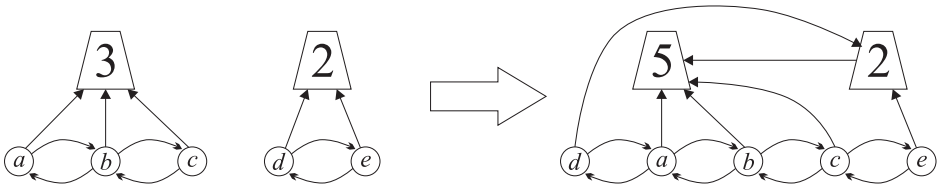

Figure 2.8: Vertex object b is currently being examined for adjacent objects during the accretion process of the edge removal algorithm. Objects a and d have already been examined, objects c and e are known to belong to their respective clumps but have not yet been examined. Suppose that b is found to be adjacent to either d or e, and so the two clumps must be merged into one. As in the edge insertion algorithm, the lesser order clump's root is made a child of the greater order clump's root, whose order counter is incremented by the lesser order. The clump to which b does not belong then has its vertex object list split, with any previously examined objects (here d) being prefixed onto b's list and any unexamined objects (here e) being affixed onto b's list. This keeps all objects that require examination to the right of b.

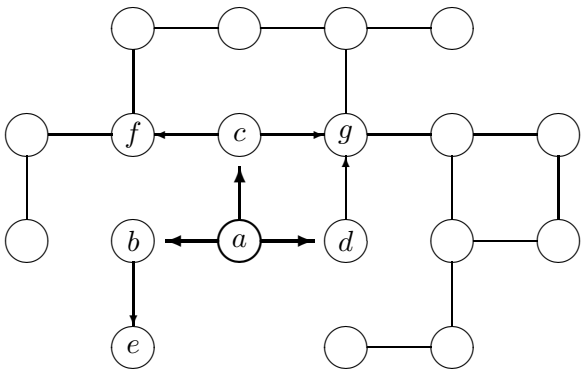

Figure 2.9: Accretion algorithm. Nucleation: vertex a and surrounding edges are removed; b, c and d are assigned to distinct clumps. Sweep one: e assigned to clump of b; f and g assigned to clump of c; clumps of g and d merged. Sweep two: clump of e complete and so is extracted to a graph; one clump remains hence all other vertices belong to the final fragment.

remains of the original graph tree is left as is, and the algorithm is complete. As shown in figure 2.9, the algorithm requires only enough time to establish the second largest fragment graph (rather than the largest), and the saving is often substantial. If clump labels are to be reused in later applications of this algorithm, then it will be necessary to make a pass over the vertex objects of this final fragment in order to remove the label from them.

Use of clump objects permits the isolation of fragment graphs without altering the original graph data structure. Since clumps are transient structures, statistics other than order need not be calculated for them. It is possible to establish the fragment graphs without using clumps or clump objects. Instead of giving vertices temporary clump labels, associated objects may be extracted directly into new graph structures. These structures may later be fused back together during the accretion process. However, this requires additional work in extracting vertex objects from the original structure and in calculating graph statistics. A significant performance difference was found in favour of using the clump approach.

An implementation that removes one edge at a time is given in script 2.4, and uses the sub-routine of script 2.5. Regarding script 2.4, the function `remove` might just as easily take the vertices as arguments rather than the edge between them. The actual deletion of the edge is a trivial matter of removing the pointers from each vertex object to the other. Clumps maintain their associated vertex objects in doubly linked lists (as per figure 2.8). Within these lists, any vertices that have already been examined in the accretion/search process are left of the `p-start` marker, and any vertices that have been added this round (and form part of the next perimeter shell, not the current one) are right of the `p-stop` marker. The final stage of this implementation, where clump data is removed, is not actually required unless clump labels are to be recycled between edge removals (which, due to memory constraints, they normally would be). In the loop where completed clump trees are turned into graph trees, the operations concerned are essentially the `consolidate` and `extract` functions of scripts 2.3 and 2.2, but the form given is more efficient here where graph structure integrity need not be maintained. The `rootof` function is defined in script 2.1. Root clump objects are stored in a ring, which is simply a doubly linked list with the ends joined together. Details of the numerous simple pointer operations required for list and ring management have been omitted.

```
define remove(edge):
    % Create a new clump upon each effected vertex
    for each vertex connected to the edge
        create new clump
        place clump in ring
        set vertex's c-parent equal to clump
        set clump's mass to unity
        set clump's p-start equal to vertex
        set clump's p-stop equal to vertex
    set activeclumps to two
    % Delete the edge connecting the two vertices
    delete edge
    % Accrete shells of vertices upon each clump (multiple breadth
    % first searches performed in parallel for all clumps)
    while activeclumps is greater than one
        advance to next clump in ring
        set swept to false
        % Scan the clump perimeter for non-member vertices
        while swept is not true
            for every vertex connected to the clump's p-start
                set clump equal to merge(clump,vertex)
            if (activeclumps is less than two) or ...
              (clump's p-start is equal to clump's p-stop)
                set swept to true
            else
                advance clump's p-start to the next vertex in the clump's list
        % This scan complete.
        if activeclumps is greater than one
            set clump's p-stop to the last vertex in the clump's list
            if clump's p-start is not equal to clump's p-stop
                % Clumps still disjoint and growing, prepare for next scan.
                advance clump's p-start to the next vertex in the clump's list
            else
                % Clump has stopped growing; turn it into a graph
                create new graph
                for every vertex in clump's list
                    set vertex's c-parent to null
                    remove vertex from list
                    set root equal to rootof(vertex)
                    set vertex's parent equal to graph
                    increment graph's mass by unity
                    decrement root's mass by unity
                remove clump from ring
                set activeclumps to one
    % Remove any ramining clump data
    if activeclumps is not zero
        for every clump in the ring
            for every vertex in the clump's list
                set vertex's c-parent to null
                remove vertex from list
    return
```

Script 2.4: Single-edge removal algorithm. In essence, two breadth first pathway searches through the graph, performed in parallel. The search is abandoned once the two clumps are established as being either connected or disconnected.

```
define merge(clump,vertex):
    set root equal to crootof(vertex)
    if (root is not equal to clump) and (activeclumps is greater than one)
        if root is equal to vertex
            % Vertex is not in any clump, add it to this one.
            append vertex to clump's list
            set vertex's c-parent equal to clump
            increment clump's mass by unity
        else
            % Vertex belongs to some other clump, consolidate the two.
            decrement activeclumps by one
            if activeclumps is greater than one
                set active equal to clump
                set passive equal to root
                divide passive list into swept & unswept sub-lists
                % (The list splits on the immediate left of p-start)
                append active's list to swept sub-list
                append unswept sub-list to active's list
                % Make smaller clump a sub-tree of the larger
                if active's mass is less than passive's mass
                    increment passive's mass by active's mass
                    set passive's p-start equal to active's p-start
                    set passive's p-stop equal to active's p-stop
                    set active's c-parent equal to passive
                    set clump equal to passive
                else
                    increment active's mass by passive's mass
                    set passive's parent equal to active
    % The clump label (root object) may have changed; hence
    % return the new label for continuity of the perimeter sweep.
    return clump
```

Script 2.5: Clump-merging subroutine for the edge removal algorithm. The above implementation is more general than required for the single-edge case of script 2.4 (where lists need never be spliced), and describes the procedure of figure 2.8. The `crootof` function is identical to the `rootof` function except that the vertex's clump tree is traversed (tracing `c-parent`s rather than `parent`s), rather than the graph tree (hence `root` is a clump (or vertex) object). The variable `activeclumps` is global to both `remove` and `merge`. In practice, the list splitting procedure involves a page of pointer operations and exception catching.

2.5 System Observation Methods

In many applications, the graphs will consist of sets of pathways-connected vertices occupying fixed positions within some lattice or other space. In this case the graphs are often referred to as clusters and their embedding within the space introduces an element of geometry to the problem. It is also common practice to refer to the order of a cluster as its mass. Below are three simple methods for calculating the spatial spanning properties of a cluster, counting the number of connected graphs present, and locating all vertices belonging to a specified connected graph.

2.5.1 Spanning Properties Method

If the graphs exist within a hypercubic space, or some other space with well defined boundary surface sectors, then, by including spatial coordinate data upon each vertex object, it becomes straightforward to establish which, if any, of the boundary sectors a given graph is in contact with. Each graph object is given a set of n counters, one for each boundary sector. The simplest graph consists of a single vertex and no edges. This is represented by a graph structure with a single (root) graph object having a single child vertex object. Spatial data upon the vertex object is examined and for every boundary the vertex is in contact with the corresponding counter is incremented upon the graph object. More complex graphs arise from the application of the algorithms of section 2.4. Whenever any of those algorithms are used, boundary data is transferred between vertex objects just as for graph order. Whenever a vertex is extracted from a graph structure, the pertinent counters upon the root object are, of course, decremented rather than incremented. In this way the root object of a graph structure always contains correct data for the number of vertices within the graph that are in contact with each of the spatial boundary sectors. This is much simpler than calculating a geometrical bounding box for each graph.

Regarding the implementation of this method, it was found (within the the context of the application of chapter 3) that it was much faster (around thirty percent or so) to supply each vertex with spatial coordinates and use conditional statements to adjust counter data upon the graph objects, than it was to supply each vertex with its own set of counters (being one if the vertex lies on the appropriate boundary and zero otherwise) and simply add

or subtract counter arrays between objects. Specifically, on a contemporary PC, the algorithms required 382 seconds to make ten-thousand sweeps of the critical region on the two-dimensional $L = 2048$ square lattice (see table 3.5) when spatial conditionals were used, and 503 seconds to perform the same task using boundary counters. This is believed to be because, although the former looks more complex in C, the resulting assembler code can perform conditionals (branch on zero, for instance) much faster than integer additions, and the vast majority of the time the number being added is zero.

2.5.2 Graph Counting Method

To keep track of the number of graphs within the system, another counter is used. Whenever a graph is created, increment the counter. Whenever two disconnected graphs become connected by an edge, the counter is decremented. Whenever an edge is removed from some graph, first decrement the counter (for the original graph) then increment the counter once for each resulting fragment graph. In this way the total number of graphs within the system is always known (this saves having to count the list of all root objects).

A simple modification of this method is to introduce a set of counters pertaining to the number of graphs having certain spanning properties (as per subsection 2.5.1). Whenever graphs are altered, the appropriate counters are identified from spanning data upon the root and updated accordingly. This method tracks information on the number of graphs of each spanning class, as well as the total number of graphs present, and was used extensively in chapter 3 for sampling data from Monte Carlo generated lattice configurations. Use of spanning counters (rather than testing whenever need be) made a small reduction in the wall clock run time for the task described in subsection 2.5.1 (from 393 seconds to 382 seconds).

2.5.3 Vertex Listing Method

By maintaining all vertex objects of a graph structure within a doubly-linked list, just as is done within clump structures (see figure 2.8), it becomes possible to quickly identify all vertex objects within the same structure. It may be preferable to maintain a pointer from the root object to the first

member of the list so that no first vertex object need be found from which to find the others (this is easily done). Such a list requires more memory but is useful if, for example, two-point spatial correlation functions are to be calculated for graph membership [126, 137, 237].

Indeed, it is actually possible to do away with the graph and clump structures of section 2.3 and use only doubly-linked lists of vertex objects to represent graphs (each list may be associated with a single root object to which every member has a pointer and is used both as a graph identity label and a storage space for graph property data). However there will always be some performance reduction with such structures since connecting two disconnected graphs with an edge now requires changing pointers (or labels) on multiple vertex objects rather than upon a single root object. This method is actually not too much slower in the context of the application of chapter 3 where all graph structures tend toward a single (root) graph object with many (direct child) vertex objects anyway. Specifically, for the task described in subsection 2.5.1, use of data lists instead of data trees was found to increase execution time from a little under 400 seconds to a little under 600 seconds. This 3:2 ratio did not appear to change much with sweep range or system size.

2.6 Summary

A data structure has been introduced that represents graphs by an underlying equivalent structure (vertices connected by edges) with a superimposed tree structure for handling graph labelling, relabelling and data tracking. A collection of algorithms has been presented for performing simple operations upon and within these structures. Because the edge insertion and edge removal algorithms both work with this common data structure, they complement each other and enable the arbitrary modification of any graph. These algorithms are themselves based upon the breadth-first search, weighted union, and find with path compression algorithms. They are designed for optimal speed and require complex data structures that occupy relatively large amounts of computer memory. This chapter has not included any demonstration of the utility of these algorithms. For applications see chapters 3 and 4.

Complementary algorithms for graphs and percolation

Michael J. Lee
Department of Physics and Astronomy, University of Canterbury, Christchurch, New Zealand
(Received 15 January 2007; revised manuscript received 3 May 2007; published 27 August 2007)

A pair of complementary algorithms are presented. One of the pair is a fast method for connecting graphs with an edge. The other is a fast method for removing edges from a graph. Both algorithms employ the same tree-based graph representation and so, in concert, can arbitrarily modify any graph. Since the clusters of a percolation model may be described as simple connected graphs, an efficient Monte Carlo scheme can be constructed which uses the algorithms to sweep the occupation probability back and forth between two turning points. This approach concentrates computational sampling time within a region of interest. A high-precision value of $p_c=0.592\,746\,03(9)$ was thus obtained, by Mersenne twister, for the two-dimensional square site percolation threshold.

DOI: 10.1103/PhysRevE.76.027702 PACS number(s): 02.70.−c, 02.10.Ox, 05.10.Ln, 64.60.Ak

The various percolation models are well-studied topological problems of statistical physics. Beyond an intrinsic fundamental mathematical importance, percolation theory boasts a number of diverse applications in areas such as environmental science, biology, geology, chemistry, engineering, physics, and cosmology [1–3].

Contemporary Monte Carlo studies of percolation typically require that large numbers of samples be taken from within the critical region of the phase transition. In order to take these samples a method such as that of Gould and Tobochnik [2], Machta, Choi, Lucke, Schweizer and Chayes [4,5], or Newman and Ziff [6,7] is often employed. These schemes all begin with an empty lattice and proceed to occupy individual sites (or bonds), one by one, until a spanning cluster exists. A high rate of sampling is achievable, but the majority of the data thus acquired are sourced from lattice configurations that lie outside the critical region. Such undesirable sampling is manifest as a serious computational inefficiency that would be better avoided.

Here a method is introduced that allows for individual sites (or bonds) to be switched back and forth between the occupied and unoccupied states. This enables the lattice to take a random walk through configurations that dwell entirely within the region of interest. Consequently a higher proportion (potentially all) of the sampled data usefully contributes to the final result.

This method relies upon two efficient algorithms for performing operations upon graphs. The first algorithm adds edges to graphs and graphs together. The second algorithm removes edges from graphs and splits graphs apart. The scope of these algorithms is much more general than merely percolation. The two algorithms share a common tree-based representation for graphs, which enables them to work efficiently with the same data structures; easily inserting and deleting both edges and vertices. While they are both applicable to general directed graphs, only simply connected undirected graphs will be considered here.

Connected graphs are represented by trees of vertex objects and graph objects (see Fig. 1). Vertex objects represent vertices; graph objects serve as a dynamic Hoshen-Kopelman relabeling table [1,8] for the vertex objects and also maintain statistics (such as order) for the graph. Each vertex object contains a pointer to the graph object of which it is a child. Likewise, each graph object contains a pointer to some other graph object of which it is a child. Trees are constructed in such a way that for each tree there always exists one, and only one, graph object which is not the child of any other object and which has no pointer. This unique object is designated as the root of the tree. Vertex objects are always leaves of the tree; graph objects never are. All vertex objects within a given tree are members of the same graph. However, it is not true, other than for the root, that the set of all vertex objects belonging to the subtree below a given graph object correspond to the vertices of a connected subgraph.

The simplest possible graph consists of a single vertex and no edges. To represent such a graph, a graph object is created without a pointer, and a vertex object is created with its pointer directed at the graph object. More complex graphs arise when the vertices of these simplest graphs become connected by edges. Vertex objects carry (additional) pointers to any other vertex objects to which they are adjacent, and these pointers represent directed edges. Undirected edges are represented by pairs of directed edges. Edges are inserted into (or removed from) a graph simply by creating (or destroying) the corresponding pointers.

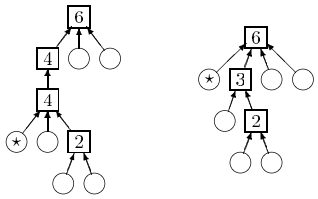

FIG. 1. The left-hand tree of graph objects □ and vertex objects ○ represents a graph. Data upon each graph object track the total number of vertex objects below. From the vertex object marked ⋆, the root is found by a recursive algorithm which follows pointers to the root and turns every object along the way into a child of the root. The result is the right-hand tree, which is a simpler representation of the same graph. The graph object that was a child of the root in the left-hand structure has been pruned from the tree.

When two vertices from two different graphs become connected by an edge, the two graphs must be fused into a single graph. This is achieved by creating a pointer to the root of the greater order graph, upon the root of the lesser order graph, thereby making the lesser graph's root a child of the greater graph's root and the lesser graph's tree a subtree of the greater graph's tree. Statistics upon the one remaining root are incremented by those on the former root, so that the data upon the remaining root correctly reflect the properties of the new graph. The result is a single tree with a unique root.

In order to perform this operation, it is necessary to locate the root of each graph tree. For every vertex object within the tree, there exists a sequence of graph objects ending with the root. For every object within the sequence, the pointer upon that object uniquely specifies the next object in the sequence. Starting with a vertex object, the sequence is found by tracing pointers from child to parent until the root is reached.

A path compression technique is applied throughout the tracing process by redirecting the pointer of every object in the sequence (other than the root) to the location of the root. In this way, all objects in the sequence are made children of the root, and future tracing operations from those objects will find the root after following only a single pointer. This affords a significant improvement in computational efficiency [6,7]. To maintain correct statistics upon each graph object, relevant information is transferred from child to parent on every step of the tracing. After path compression, some graph objects from the sequence may have no children. These redundant objects are pruned from the tree and deleted from memory during the tracing process. The complete root finding process is shown in Fig. 1.

An edge may be removed from between two vertices by deleting the appropriate pointers from the corresponding vertex objects. A vertex may be removed from a graph by removing all of the edges about that vertex (the associated vertex object may then be deleted). Such operations may cause the original graph to fragment into a set of smaller graphs. The deterministic accretion algorithm which efficiently identifies these fragments is similar to the stochastic methods of Hammersley and Handscomb [9], Leath [10], and Alexandrowicz [11], where sites and bonds are added to the perimeter of an existing percolation cluster.

Each surviving vertex that has had one or more of its edges removed is assigned to a distinct clump. A clump is merely a label held in common by a set of vertices that are all known to belong to the same fragment graph. Each such labeled vertex forms a separate nucleation kernel for an accretion process which constructs the fragment graphs from the remains of the original. From each labeled vertex, all edges are followed outward to what must be an adjacent vertex, and these are assigned to the same clump as the labeled vertex. As vertices are added to the perimeter of a clump, they are placed in a queue to be later examined for adjacent vertices that also need to be added to the clump. This breadth-first process is performed in parallel for all clumps, and was found to be significantly faster than depth-first accretion.

When two adjacent vertices are found to belong to different clumps, those two clumps are merged into one. When a clump contains no vertex that is adjacent to any other vertex that is not a member of the same clump, then that clump has stopped growing and must represent a complete connected fragment graph. All vertex objects associated with the clump are then extracted from the original graph tree and are made children of a newly created (root) graph object. Parameters for the fragment graph are calculated upon the root object, and the clump ceases to be.

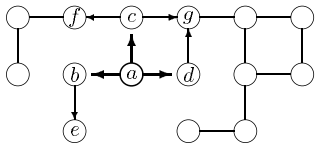

FIG. 2. Accretion algorithm. Nucleation: vertex a and surrounding edges are removed; b, c, and d are assigned to distinct clumps. Sweep 1: e assigned to clump of b; f and g assigned to clump of c; clumps of g and d merged. Sweep 2: clump of e complete and so extracted to a graph; one clump remains and hence all other vertices belong to the final fragment.

When the number of distinct clumps remaining drops to 1, whatever remains of the original graph tree must logically represent the final fragment graph. It is not necessary to continue with the accretion process, the original graph tree is left as is, and the algorithm is complete. As shown in Fig. 2 the algorithm requires only enough time to establish the second largest fragment graph, rather than the largest, and the saving is often substantial.

Clump labeling is achieved by the use of clump objects to which vertex objects may or may not have pointers, exactly as in the graph trees. The same pointer method is used to merge clumps as is used to fuse graphs. The same path compressing pointer tracing process is used to find clump labels (clump tree root objects) as is used to find graph tree roots. Use of clump objects permits the isolation of fragment graphs without altering the original graph tree data structure. Since clumps are transient structures, statistics other than order need not be calculated for them.

It is possible to establish the fragment graphs without using clumps or clump objects. Instead of giving vertices temporary clump labels, associated objects may be extracted directly into new graph trees. These trees may later be fused back together during the accretion process. However, this requires additional work in extracting vertex objects from the original graph tree and in calculating graph statistics. A significant performance difference was found in favor of using the clump approach.

Vertices can represent the binary state sites of a percolation model. Graphs become equivalent to clusters of occupied sites simply connected by occupied bonds. Unoccupied sites have neither graph object nor clump object pointers upon their corresponding vertex objects, and are not members of any graph or clump. An occupied site belongs to a cluster uniquely identified by the root of the graph tree to which the corresponding vertex object belongs. Bonds may be occupied or unoccupied by switching edge pointers on and off upon the sites. The algorithm presented here for join-

ing two graphs becomes similar to the site-to-site pointer tree method of Newman and Ziff [6,7]. It is not entirely equivalent, since it is impractical to do away with the graph objects while still retaining the edge and vertex removal component of the method.

Consider a site percolation model upon a fixed lattice of N sites. The pair of algorithms presented here confer the ability to arbitrarily raise and lower the number of occupied sites n upon the lattice while efficiently maintaining the correct cluster information and associated statistics. The algorithms may be forged into a Monte Carlo scheme which sweeps n back and forth between two turning conditions. Within this scheme, a Monte Carlo step consists of either the occupation of a randomly chosen unoccupied site, or the deoccupation of a randomly chosen occupied site. Initially, n is stepped upward until the upper turning condition is satisfied, and then n is stepped downward until the lower turning condition is met. This cyclical process may be repeated indefinitely. This (bidirectional sweeping) approach is a generalization of the single stopping condition (and implicit starting condition at $n=0$) found in the (unidirectional sweeping) algorithms of Machta *et al.* [4,5] and of Newman and Ziff [6,7]. While this method is conceptually simple, its computational performance depends strongly upon the details of how the Monte Carlo steps are realized. The intention of the complementary graph algorithms described here is to achieve these steps in as little time as possible.

The turning conditions might be a change in the order parameter of the system. The occupation n can be increased until a spanning cluster exists, and then decreased until a spanning cluster no longer remains. The result is a self-organized critical random walk through lattice configurations which are all only a single Monte Carlo step away from the phase change. Such an approach hints at that of Tomita and Okabe [12], where the current measured value of some order parameter determines a change in bond density.

In order to measure the square site percolation threshold p_c upon an $N=L\times L$ lattice, an unbiased Monte Carlo estimator of the spanning probabilities $R_L(p)$ [13,14] is useful. The occupation probability $p=n/N$. To provide this unbiased estimate, the turning conditions are taken to be two fixed values of n, so that all configurations of a given n are visited with equal probability. Since knowledge of $R_L(p)$ is required only about the critical point, the turning points are chosen such that the sampling range includes this region.

Here lies an advantage over earlier unidirectional sweeping algorithms which return information for all n from zero to the phase transition, a range of $O(N)$ steps. The critical region spans a smaller range of only $O(NL^{-1/\nu})$ steps [1]. In two dimensions the correlation length exponent $\nu=4/3$ [1,2] and so the range is $O(N^{5/8})$. Restricting the sampling range to the critical region significantly reduces the fraction of samples that make no contribution to the final result. While the method presented here does not require any *a priori* knowledge, it is able to make use of any information that does exist. The performance and sampling range may be adjusted accordingly.

Estimating $R_L(p)$ requires knowing of the existence, or otherwise, of lattice spanning clusters. An efficient means for tracking this information is integrated into the tracing process of Fig. 1. Each vertex object carries data indicating on which boundaries of the lattice the associated site lies. When a site becomes occupied, these data are transferred to the root graph object where a running total is kept for all sites within the associated cluster. It is then straightforward to determine which dimensions of the lattice are spanned by this cluster, and that information is in turn sent from the root object to a master array for the entire lattice. This array is updated whenever a cluster is modified, so that at any given instant the precise number of clusters spanning any given combination of lattice dimensions is known.

Define $s_d(n)$ as the number of occasions on which the lattice is found to be in a configuration at occupation level n within which there exists a cluster that spans the lattice across d, or more, dimensions. Consequently, $s_0(n)$ is the total number of observations made at occupation level n and, in two dimensions, $s_2(n)$ is the number of those observations in which a single cluster spans across the entire lattice in both dimensions. The number of observations in which a cluster spans one, and only one, (unspecified) spatial dimension is given by $s_1(n)-s_2(n)$. After making numerous observations, the probability that the lattice will be spanned over a given (specified) spatial dimension by a randomly generated configuration at occupation level n is estimated by

$$R_L(n/N) = \frac{s_1(n) + s_2(n)}{2s_0(n)}.$$

It has been shown [13,14] that for large L the spanning probability at the critical point is given by

$$R_L(p_c) \approx 0.5 + b/L.$$

Ziff and Newman have found that $b=0.320(1)$ [15]. Working on a lattice of $L=2048$ ($N=4\,194\,304$), it follows that $b/L=0.000\,156\,25(50)$.

Calculations were performed to make estimates of $R_L(p)$. Two sampling ranges were used; the entire critical region, $2\,474\,000 \leqslant n \leqslant 2\,498\,000$, and a small neighborhood about the critical point, $2\,485\,700 \leqslant n \leqslant 2\,486\,700$. In both cases the decorrelation time was found to be on the order of 5×10^4 steps. This compares favorably with the 2.5×10^6 steps required to take a sample by unidirectional methods from an initially empty lattice.

Preliminary trials were conducted with a decimated Mitchell and Moore additive lagged Fibonacci generator with taps at 418 and 1279 [16]. Results were consistent between using a single such generator and using an independently seeded pair to obtain x-y site coordinates from only the most significant bits of each. A combined total of 3×10^9 range sweeps yielded an estimate of $p_c=0.592\,746\,63(24)$.

High-precision measurements were conducted with Matsumoto and Nishimura's Mersenne twister generator MT19937 [17]. Results were consistent between using a single generator and using a decimated independent x-y pair. A total of 2×10^{10} range sweeps produced the estimates $R_L(2\,486\,156/N)=0.500\,097(22)$, $R_L(2\,486\,157/N)$

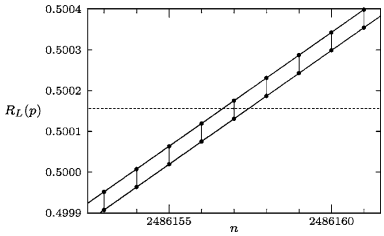

FIG. 3. Combined Mersenne-twister-based Monte Carlo estimates for the probability of the existence of a cluster spanning a single specified lattice dimension of a square site percolation model with $N=2048^2$ sites. Solid lines are the error bounds for $R_L(p)$, the dashed line is $R_L(p_c)$.

$=0.500\,153(22)$, and $R_L(2\,486\,158/N)=0.500\,209(22)$, as shown in Fig. 3.

It follows that the best estimate for the square site percolation threshold is $p_c=0.592\,746\,03(9)$, a measurement of significantly greater precision than earlier results [6,13,18] (see also Ref. 19 of [6]), and which should assist with future studies of the model. Note that this value is based purely upon the Mersenne twister calculations and differs from the lagged Fibonacci estimate. At this level of precision, the choice of pseudorandom number generator is clearly of great importance.

In summary, efficient deterministic algorithms have been presented for the manipulation of graphs. These are potentially useful in topological problems such as the analysis of networks and perturbative expansions in diagram formalisms. It has been shown how they form the basis of an efficient Monte Carlo method which walks a system through every point within a chosen range and, crucially, only those points. By exclusively sampling from pertinent system configurations, improved computational efficiency is achieved. Greater quantities of useful data can be acquired than with other techniques, and the result is an increased precision of measurement. The method was used to achieve the most precise estimate to date of the square site percolation threshold and is applicable to a wide variety of numerical experiments on any discretized space of arbitrary connectivity.

Acknowledgements are due to O. K. L. Petterson and C. J. McMurtrie for the provision of computing resources, and to R. M. Ziff for useful information and suggestions. Elements of this work were conducted upon the University of Canterbury Supercomputer.

[1] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, revised 2nd ed. (Taylor and Francis, London, 1994).
[2] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods*, 2nd ed. (Addison-Wesley, Reading, MA, 1996).
[3] *Nearby Large Scale Structures and the Zone of Avoidance*, edited by A. P. Fairall and P. A. Woudt (Astronomical Society of the Pacific, Provo, UT, 2005).
[4] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. V. Chayes, Phys. Rev. Lett. **75**, 2792 (1995).
[5] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. M. Chayes, Phys. Rev. E **54**, 1332 (1996).
[6] M. E. J. Newman and R. M. Ziff, Phys. Rev. Lett. **85**, 4104 (2000).
[7] M. E. J. Newman and R. M. Ziff, Phys. Rev. E **64**, 016706 (2001).
[8] J. Hoshen and R. Kopelman, Phys. Rev. B **14**, 3438 (1976).
[9] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods* (Methuen, London, 1964).
[10] P. L. Leath, Phys. Rev. B **14**, 5046 (1976).
[11] Z. Alexandrowicz, Phys. Lett. **80**, 284 (1980).
[12] Y. Tomita and Y. Okabe, Phys. Rev. Lett. **86**, 572 (2001).
[13] R. M. Ziff, Phys. Rev. Lett. **69**, 2670 (1992).
[14] P. J. Reynolds, H. E. Stanley, and W. Klein, Phys. Rev. B **21**, 1223 (1980).
[15] R. M. Ziff and M. E. J. Newman, Phys. Rev. E **66**, 016129 (2002).
[16] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 2nd ed. (Addison-Wesley, Reading, MA, 1981).
[17] M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).
[18] Y. Deng and H. W. J. Blöte, Phys. Rev. E **72**, 016126 (2005).

Chapter 3

The Percolation Threshold

An introduction to classical percolation theory is followed by a review of techniques commonly employed in its study. New methods, using the graph algorithms of the previous chapter, are then introduced. These are subsequently applied to a specific long-standing problem; the accurate determination of the square site percolation threshold to a high level of precision.

3.1 The Percolation Model

Percolation is the science of connectedness [246, 22, 99, 178, 51]. Percolation theory provides a quantitative description of the nature of continuous pathways through space. The usual objective of percolation research is to characterise some aspect of the critical phenomena of the phase transition between finite and infinite range connectivity. This is non-trivial when the space is randomly disordered and the connected regions acquire fractal properties [301, 166, 242, 165].

In the language of percolation, connected regions are called clusters. A cluster may be defined as an irreducible pathwise-connected topological space. Pathwise-connected is used in the usual sense that for every pair of points from within the space there exists a path connecting them [273] (such a path may or may not pass through intermediate points of the space). Hence every element of the space is accessible from every other element of the space. Irreducible is used to indicate that any point that is accessible from any point within the space must itself be a member of the space.

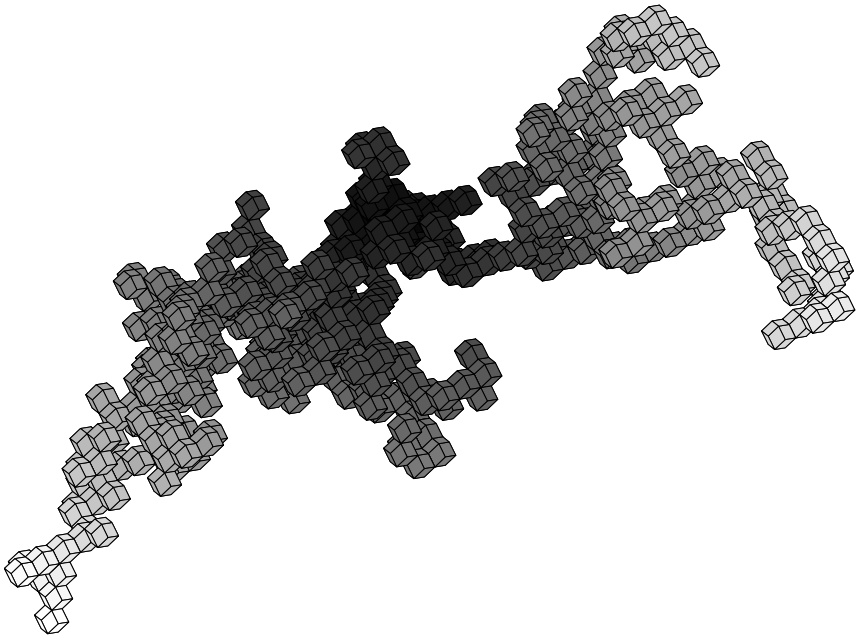

Figure 3.1: A three-dimensional cluster grown by Leath algorithm upon the face-centred cubic lattice near its critical point (refer table 3.1 and subsection 3.3.4).

An example of a cluster is shown in figure 3.1. This particular object is composed of space-filling polyhedra (rhombic dodecahedra) whose centres lie upon a face-centred cubic lattice embedded within Euclidean three-space.

It is from such embeddings that purely topological entities acquire the additional complexity of geometry. In the case of embedded clusters, it becomes sensible to ask questions of shape, density and surface area, amongst others. When the clusters are constructed stochastically, their geometrical properties tend to be statistically fractal and conform to (non-integer) power-law distributions [166, 242]. With embedding comes also the possibility of multiple clusters holding various relative positions within a common metric space that they cohabit. A specific arrangement of this type is commonly referred to as a configuration or pattern.

Clusters are the fundamental objects of study in percolation. While the clusters are compound entities, consisting of a collection of elements of the embedding space, they have properties, both individually and collectively,

that are independent of those underlying elements. As the density of (randomly placed) cluster elements increases within the metric space, the clusters will tend to grow in size and merge together. The clusters remain finite in extent (and are referred to as bounded or localised) up to some particular value of the density. At that value a phase transition takes place where an infinite number of finite size clusters gel together into a single infinite size cluster. This infinite cluster of unbounded extent is called the percolating cluster and the density at which the gelation takes place is known as the percolation threshold. The percolation threshold marks the sharp boundary between localised and extended connectivity. The numerical value of the threshold is dependent upon the topology of the specific percolation model in question.

Amongst the most commonly studied models in percolation are those on regular lattices [246, 249]. Lattices are spatially discretised and topologically non-compact. Topologically non-compact spaces consist of a set of isolated points and a set of connections (pathways) between them. In the language of percolation theory, points are called sites and connections are called bonds. Sites and bonds exist in either of two possible states, normally called occupied and unoccupied respectively. Clusters are typically defined as irreducible topological spaces of occupied sites, pathways-connected by occupied bonds. In some cases, where geometric dual configurations are also of interest, it may be useful to include irreducible spaces of unoccupied sites connected by unoccupied bonds within the definition of cluster. Two forms of the lattice percolation model, known as site percolation and bond percolation, are usually identified.

In the bond percolation model, the state of each bond on the lattice is chosen independently and randomly. States of the sites are determined from those of the adjoining connecting bonds. There are different ways of doing this depending upon the properties required of the model and its geometric dual, but the minimum condition is that if a site has one or more occupied bonds about it then the site is also occupied. A common choice is simply to make all lattice sites occupied. An example of bond percolation on a square lattice is shown in figure 3.2. In this case, a spanning cluster exists where it is possible to travel between opposite lattice boundaries via a walk on this single cluster. Such a configuration is in the supercritical or spatially extended phase. The spanning cluster is the finite sized lattice analogue of

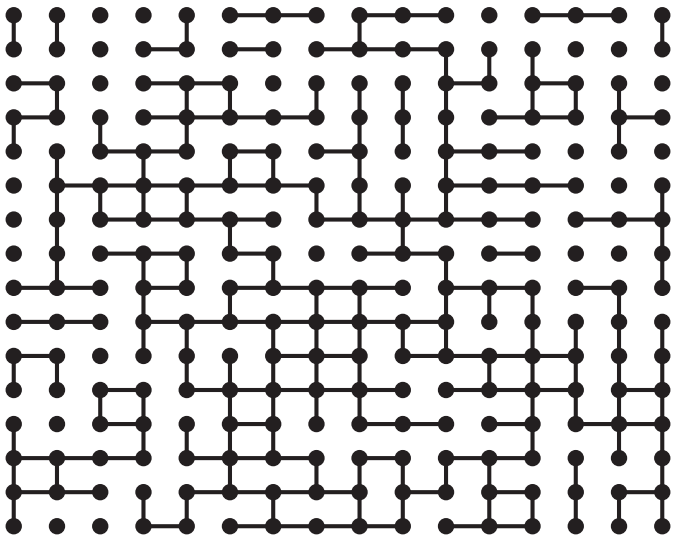

Figure 3.2: Sample configuration of a classical bond percolation model on the square lattice. There are forty clusters present, including a single (spatially extended) spanning cluster that streches completely across the lattice.

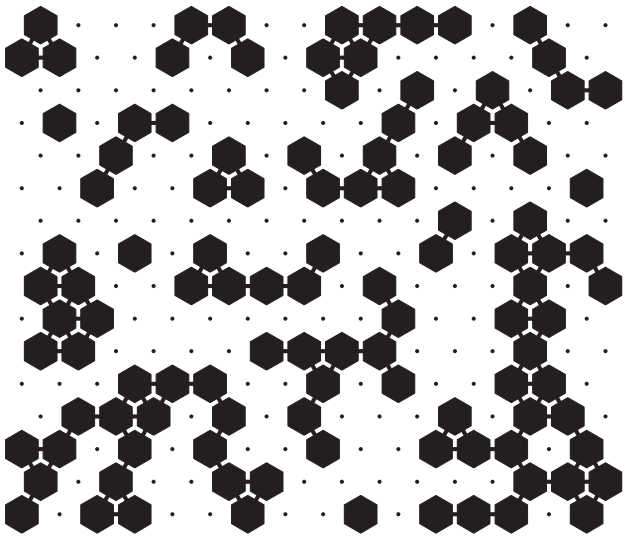

Figure 3.3: Sample configuration of a classical site percolation model on the triangular lattice. There is no spanning cluster present, and hence this system is in the subcritical (spatially localised) phase. Note that this lattice has a duality property where if no spanning cluster of occupied sites exists, then a (dual) spanning cluster of unoccupied sites necessarily does.

the percolating cluster on the infinite lattice.

In the site percolation model, the state of each site on the lattice is chosen independently and randomly. States of the bonds are determined from those of the sites at either end. There are different ways of doing this depending on the properties required of the model and its geometric dual, but the minimum condition is that any bond between two occupied sites must itself be occupied. An example of site percolation on a triangular lattice is shown in figure 3.3 (occupied sites are solid). In this case, no spanning cluster exists and only a limited range of travel is possible from any given starting point. Such a configuration is in the subcritical or spatially localised phase.

Often the objective in studying some or other percolation model is to make a quantitative determination of some aspect of its critical phenomena [293, 68, 18]. This generally involves an analysis of the statistical distribution of some property of the clusters at or near the percolation threshold. The threshold itself is, of course, defined by the existence or otherwise of the infinite cluster. Important early review articles on the subject include those of Frisch and Hammersley [75], Stauffer [245], Essam [62], and several articles within the collection edited by Deutscher, Zallen and Adler [51] (particularly those by Zallen [284], Domb [54], and Hammersley [104]). The standard introductory text is by Stauffer and Aharony [246].

Besides having its own mathematical interest [69, 260], percolation theory lays claim to a large number of diverse applications [284, 222]. Traditionally these have been associated with the spatial properties of disordered physical and chemical systems. However, many recent applications derive from percolation theory's usefulness in the study of complex networks, and extend well beyond the hard sciences. Known applications include cosmology and astrophysics [65, 222, 228, 230, 64], gaseous matter physics [42], condensed matter physics [25, 108, 50, 14, 97], the Ising and Potts models [250, 278, 164, 163, 263], diffusion and transport [246, 181, 19, 235, 135, 194], the Hall effect [127, 14, 264, 150, 56], gelation [124], universal critical phenomena [244, 247, 235], chemical kinetics [70, 23, 83], electrical engineering [6, 209], forestry [111, 246] (see figure 3.4), biological evolution [125], the spread of disease epidemics [184], marketing [85], commercial collaborations [189], social networks [241], opinion dynamics [255], influence of the mass media [205], the entertainment industry [189], airline route planning [27], the European rail network [143], the human brain [143], electrical power grids

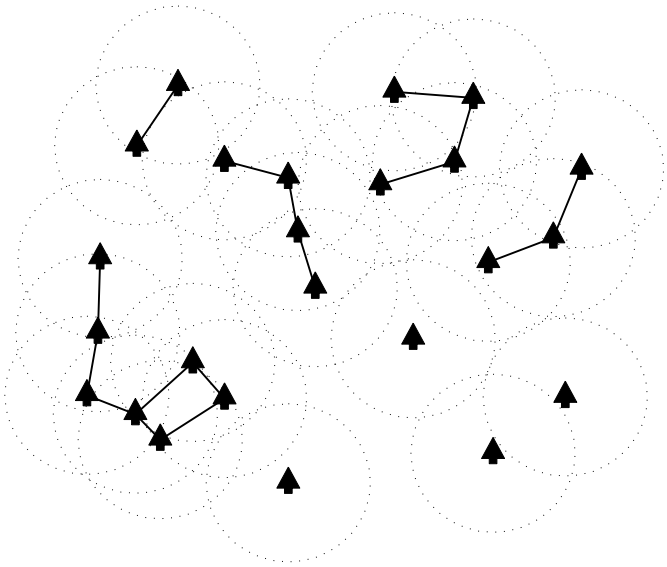

Figure 3.4: A continuum percolation sparse forest fire model. If the flames can only jump a certain distance between trees, then the fire can only spread to trees in the same cluster.

and infrastructure [27], communication networks (primarily the internet) [27, 189], geological rock mechanics [267, 221], oil recovery [246, 144, 155, 32], and to scientific collaborations of the researchers themselves [189].

Clearly, percolation models are not short for applications and it was observed that over the last quarter of the past century roughly four thousand papers had been published with the word 'percolation' appearing in the title [190, 245]. A quick search on the Web of Science database confirms this observation and produces the data shown in figure 3.5. The earliest origins of the subject coincide with the advent of computers, since percolation was developed partially as a problem for which computers would be useful [104, 246]. With tongue-in-cheek, one might then be tempted to say that a phase transition occurred in the field around the year 1980, give or take. Indeed there is some truth to this, for the elegantly simple models are, in practice, intractably difficult to solve by analytical means and the field has become intimately linked with Monte Carlo simulation. The increased availability of the personal computer around this time therefore made the study of percolation problems much more widely accessible. Computer availability

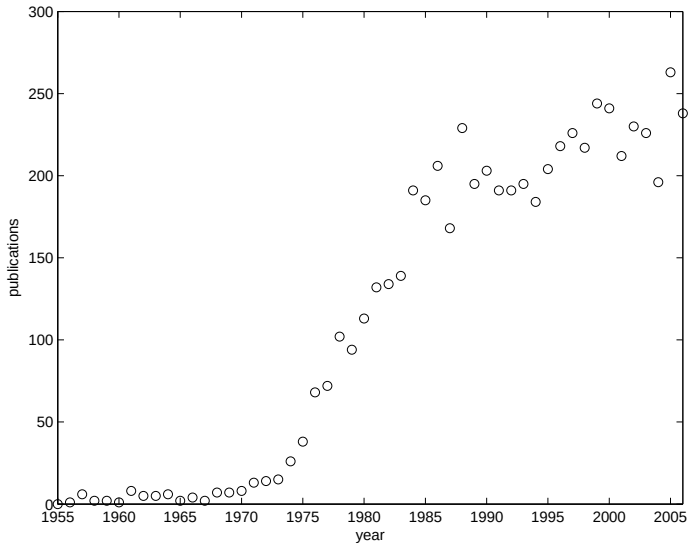

Figure 3.5: Number of papers appearing annually with the word 'percolation' in the title (search results from the Web of Science database). This figure is essentially an update of that due to Stauffer [245].

tends to remain a limiting factor upon the work that gets accomplished and hence there is a market for more efficient methods to maximise return.

As has been observed by many authors, including Stauffer and Aharony [246], Sahimi [222], Bollobás and Riordan [22], and Ziff [293], percolation problems are typically conceptually simple and easily defined. The same authors then note that those same problems are almost always intractably difficult to solve. While this enhances the mathematical appeal of percolation, it also means that very few exact solutions are available. In time, the various unsolved models may well fall to exact methods (stochastic Loewner evolution being one recent promising development [227, 101]). However, it is worth noting that the square site lattice, in particular, lacks the geometric duality symmetry property that been responsible for almost all proven analytical values of percolation thresholds [22, 251, 253, 131, 276, 298] (the only analytical result for the infinite lattice in percolation that does not follow from duality is a conjecture for the Potts model due to Wu [281, 293]). In the absence of formally proven results, high precision numerical experiments are needed to test conjectures, to inspire exact results [225, 240],

and to permit further numerical experiments (since these often rely upon an accurate threshold, as is the case for studies of excess cluster numbers [61, 245, 246]). Famously, the Alexander-Orbach conjecture for the fracton exponent [3] was numerically disproven in a series of five back-to-back papers [283, 112, 114, 207, 159]. Although Monte Carlo experiments within a computer are generally much more powerful and convenient for the study of percolation, it is still possible to conduct real-world physical experiments such as the conductivity experiment of Last and Thouless [145], and the famous Watson and Leath chicken wire experiment [271]. A small collection of thresholds for various site and bond problems is provided for reference in table 3.1. The precise determination of these values is an ongoing job, as evidenced by the recent history of estimates for the simple cubic site percolation threshold in table 3.2, and by the history of square site percolation threshold estimates in table 3.3.

So the percolation model, now over sixty years old, has proven its utility through numerous diverse applications, and yet many fundamental mathematical problems remain open. Central to many of these problems is the exact location of the critical point, and, so long as analytical solutions remain elusive, high precision numerical estimates are much sought after [249, 229, 199]. As noted by Ziff [293], these numerical problems are simple in principle, but tricky in practice and there is a need for efficiency. They are easy to do, but hard to do well. Computer resources being finite, it is desirable to have as efficient a method as possible for performing such calculations, thereby obtaining the best result possible within the limitations of the equipment and time available.

3.2 Review of Analytical Methods

Stochastic percolation models have a great deal of mathematical appeal since problems are of a geometrical nature and are simple to define yet difficult to solve [246, 222, 22, 293]. For a more detailed survey of analytical methods and results than is given in the brief overview provided here, the interested reader may wish to refer to the standard reference text by Grimmett [99] (bond percolation on the square lattice). Meester and Roy cover the more general case of continuum percolation where sites are not constrained to a regular lattice [178], and a review by Chayes and Chayes deals with percola-

Lattice	Site Threshold	Bond Threshold
Honeycomb	0.697043(3)[a]	0.652703645...[b]
Kagomé	0.652703645...[b]	0.5244053(3)[c]
Square	0.59274621(13)[d]	0.500000000...[e]
Triangular	0.500000000...[e]	0.347296355...[b]
Simple cubic	0.3116077(3)[f]	0.2488
Body-centred cubic	0.2459615(10)[g]	0.1803
Face-centred cubic	0.1992365(10)[g]	0.119
Four-dimensional hypercubic	0.196889(3)[h]	0.1601
Five-dimensional hypercubic	0.14081(1)[h]	0.1182
Six-dimensional hypercubic	0.107	0.0942
Seven-dimensional hypercubic	0.089	0.0787

Table 3.1: Percolation thresholds for various regular lattices. Values are from table 1 of Stauffer and Aharony [246] (replicated in [273]), except for: [a]Suding and Ziff [249]; [b]Wierman [274] (exact result); [c]Ziff and Suding [300]; [d]Newman and Ziff [190]; [e]Kesten [132] (exact result); [f]Deng and Blöte [46]; [g]Lorenz and Ziff [160]; [h]Paul, Ziff and Stanley [200]. Algebraic expressions for the exact results may be found in section 3.2. Mean field behaviour (rather than threshold) is expected for systems of greater than seven dimensions [46, 187].

Year	Result	Author(s)	Ref.
1992	0.311604(6)	Grassberger	[91]
1998	0.3116(1)	Lin and Hu	[157]
1998	0.311600(5)	Jan and Stauffer	[123]
1998	0.3116080(4)	Lorenz and Ziff	[160]
1999	0.3116081(13)	Ballesteros *et al.*	[11]
2003	0.3115(3)	Martins and Plascak	[175]
2005	0.3116077(3)	Deng and Blöte	[46]

Table 3.2: Recent history of site percolation threshold measurements on the simple cubic lattice (largely reproduced from Deng and Blöte [46]).

Year	Ref.	Author(s)	Method	PRNG(s)	Result
1960	[59]	Elliot *et al.*	Fifth-order series		0.48
1961	[55]	Domb and Sykes	Ninth-order series		0.55
1961	[76]	Frisch *et al.*	MC, 2000 sites		0.581(15)
1963	[45]	Dean	MC, 78^2 sites		0.580(18)
1964	[252]	Sykes and Essam	Tenth-order series		0.59(1)
1976	[254]	Sykes *et al.*	19th-order series		0.593(2)
1976	[219]	Rousenq *et al.*	MC, 1000^2 sites		0.595
1976	[243]	Stauffer	Series analysis		0.591(1)
1976	[115]	Hoshen *et al.*	MC, 4000^2 sites		0.5927(3)
1980	[212]	Reynolds *et al.*	MC, 500^2 sites		0.5931(6)
1982	[48]	Derrida and de Seze	Transfer matrix		0.5927(2)
1982	[53]	Djordjevic *et al.*	Series analysis		0.5923(7)
1984	[82]	Gebele	MC, 50000^2 sites		0.59277(5)
1985	[208]	Rapaport	MC, 160000^2 sites		0.5927(1)
1985	[49]	Derrida and Stauffer	Transfer matrix		0.59274(10)
1985	[218]	Rosso *et al.*	Gradient frontier		0.59280(1)
1986	[286]	Ziff	Perimeter walks		0.59275(3)
1986	[130]	Kertész	Transfer matrix		0.59273(6)
1986	[297]	Ziff and Sapoval	Hull-gradient	T	0.592745(2)
1988	[299, 296]	Ziff and Stell	Hull-gradient	QTA	0.5927460(5)
1989	[282]	Yonezawa *et al.*	Planar crossing		0.5930(1)
1992	[287]	Ziff	Hull-crossing	QTA	0.5927460(5)
1994	[119, 117]	Hu	Histogram MC	F	0.592(8)
1995	[120, 117]	Hu	Histogram MC	F	0.5928(1)
1996	[121, 117]	Hu *et al.*	Histogram MC	F	0.59278(2)
1996	[121, 117]	Hu *et al.*	Histogram MC	F	0.59283(4)
1996	[121, 117]	Hu *et al.*	Histogram MC	F	0.59267(6)
1996	[121, 117]	Hu *et al.*	Histogram MC	F	0.5814(30)
1996	[121, 117]	Hu *et al.*	Histogram MC	F	0.6041(30)
2000	[190, 191]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.59274621(13)
2000	[190, 191]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.59274636(14)
2000	[190, 191]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.59274606(15)
2000	[190, 191]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.59274629(20)
2000	[190, 285]	Ziff	Hull-gradient	QTB	0.5927465(2)
2002	[296]	Ziff and Newman	Planar crossing	QTB	0.5927464(5)
2003	[175, 174]	Martins and Plascak	Toroidal wrapping	C	0.5927(1)
2003	[175, 174]	Martins and Plascak	Toroidal wrapping	C	0.5929(3)
2003	[196, 195]	Oliveira *et al.*	Toroidal spanning	R	0.59274675(88)
2003	[196, 195]	Oliveira *et al.*	Toroidal spanning	R	0.59274621(33)
2005	[46, 21]	Deng and Blöte	Cylindrical correlation	TTT	0.5927465(4)
2005	[46, 21]	Deng and Blöte	Cylindrical correlation	TTT	0.5927466(6)
2005	[46, 21]	Deng and Blöte	Cylindrical correlation	TTT	0.5927466(8)
2005	[46, 21]	Deng and Blöte	Cylindrical correlation	TTT	0.5927468(10)
2005	[10]	Balister *et al.*	Semi-rigorous	MT	0.5927(8)
2007	[214]	Riordan and Walters	Semi-rigorous	MT	0.59275(25)

Table 3.3: Previously published estimates of the square site percolation threshold. The pseudorandom number generator(s) used are given where known (see subsection 3.6.1 for definitions). Generator T is a Tausworthe generator, while C is a congruential generator. R is a feedback shift register with taps at 147 and 250, seeded from the multiplicative congruential generator $x_i = 16087x_{i-1} \mod 2^{31} - 1$ (x odd). TTT is the generator most likely used by Deng and Blöte (the actual generator was definitely of this class). References are provided for both the result and the generator whenever those come from different sources. Uncertainties are quoted as one standard deviation statistical errors, except in the semi-rigorous results of Balister, Bollobás and Walters (99.99% confidence bound) and of Riordan and Walters (99.9999% confidence bound). Only those results derived from currently accepted scaling relations are shown from the greater collection in Hu, Chen and Wu (Fortran compiler dependent congruential generators F were normally used). This table is essentially a continuation of that due to Ziff and Sapoval [297].

tion in a mathematically general form [35]. Numerous recent mathematical review articles of note include those of Chayes, Puha and Sweet [36], Kesten [133], and Grimmett [78]. Recently Bollobás and Riordan [22] have compiled the most important proofs in percolation, including the Harris-Kesten result and the uniqueness of the infinite cluster. Very recently, Balister, Bollobás, Walters and Riordan have devised a method for semi-rigorous bounds on percolation thresholds [10, 214] (see table 3.3).

3.2.1 Geometric Transformations

One of the earliest mathematically rigorous results in percolation (on the infinite lattice) is a lower bound of one-half for the square lattice bond percolation threshold, established by Harris in 1960 [107]. Such formally proven results are relatively hard to come by however. This seems especially true for the critical points (refer table 3.1), and statistical physicists commonly resort to semi-rigorous or approximate methods.

One of these, introduced in 1963 by Sykes and Essam [251], is based around Kennelly's star-triangle (or Δ-Y) transformation for electrical circuits [129]. This approach uses the self-duality and self-matching properties of certain lattices (or pairs of lattices) to make percolation threshold estimates upon them [276]. Although analytical, these results are non-rigorous since the method relies on unproven assumptions [131]. The specific percolation threshold estimates obtained by Sykes and Essam were $1/2$ for site percolation on the triangular lattice, $2\sin(\pi/18)$ for bond percolation on that same lattice, $1/2$ for bond percolation on the square lattice, and $1-2\sin(\pi/18)$ for bond percolation on the honeycomb lattice [253].

The bond percolation threshold estimate for the square lattice was later confirmed by Kesten who, in 1980, used results due to Seymour and Welsh [231] and Russo [220] to show that $1/2$ is an upper bound for the square lattice bond percolation threshold [131], thereby completing the rigorous proof begun by Harris some twenty years earlier. Formal proofs confirming the remaining three conjectured values followed shortly thereafter [274, 132]. Furthermore, geometric equivalence implies that the site percolation threshold for the Kagomé lattice is identical to the bond percolation threshold on the honeycomb [300]. Recently, Ziff and Scullard have generalised the transformation, thereby enabling its application to a wider class of lattices [229, 298, 292]. However the Sykes-Essam method is not infallible as counter-

examples exist for some estimates thus made [13, 275].

3.2.2 Renormalisation Group Methods

Several real-space renormalisation group methods have been developed for percolation with the aim of providing indicative results for comparison with measured values from numerical experiments. These include statistical physics based approaches, such as that of Harris, Lubensky, Holcomb and Dasgupta [106], that draw parallels between percolation models and physical spin systems. Such methods have been used by Giri, Stephen and Grest [84], and Burkhardt and Southern [26], to suggest that the critical exponents of the site and bond models are identical. Further real-space renormalisation group methods are based purely upon probability and combinatorics. These include the methods of Reynolds, Stanley, Klein and Coniglio [210, 139, 211] and of Shapiro [234].

3.2.3 Polynomial Enumerations

Enumerative exhaustion methods are viable only for very small lattices. By considering all possible configurations, Reynolds, Stanley and Klein have derived exact polynomials for the existence probability of spanning clusters on small square lattices of up to twenty-five sites [212]. Ziff and Newman have since extended this work to lattices of up to forty-nine sites [296]. Coefficients of the polynomials were computed by a deterministic (systematic and exhaustive) variant of the hull-walk algorithm (subsection 3.3.5). Although useful, these results do not solve the problem of locating the true critical point on the infinite lattice.

3.2.4 Conjectures

The tendency for physicists to derive analytical expressions from unproven assumptions has lead to many conjectured results, and subsequent debates, within percolation. As noted in section 3.1, one of the best known is the Alexander-Orbach conjecture that the fracton exponent equals 4/3 in all dimensions [3, 246]. This was famously disproven by numerical experiment in five back-to-back papers by Zabolitzky [283], Herrmann, Derrida and Vannimenus [112], Hong, Havlin, Herrmann and Stanley [114], Rammal, Angles d'Auriac and Benoit [207], and Lobb and Frank [159]. Another example

Quantity	Relation	Exponent
order parameter	$P_\infty \propto (p - p_c)^\beta$	$\beta = 5/36$
mean size of finite clusters	$S(p) \propto \lvert p - p_c \rvert^{-\gamma}$	$\gamma = 43/18$
connectedness length	$\xi(p) \propto \lvert p - p_c \rvert^{-\nu}$	$\nu = 4/3$
cluster numbers	$n_s(p = p_c) \propto s^{-\tau}$	$\tau = 187/91$

Table 3.4: Some exact critical exponents of the percolation phase transition in two dimensions (reproduced from Gould, Tobochnik and Christian [88]).

is Ziff and Suding's [300] numerical disproof of both the (competing) Wu [280] and Tsallis [265] conjectures for the bond percolation threshold on the Kagomé lattice. Table 3.4 gives exact values for some critical exponents of two dimensional percolation. Until recently, these were merely conjectures, well supported by experimental data. They have now been proven, by Smirnov and Werner, using a stochastic Loewner evolution technique on the triangular lattice [240].

3.2.5 Cardy's Formula

As recounted by Ziff [293], an important explicit expression for the horizontal crossing probability Π_h for site percolation on a large (L tending to infinity) rectangular lattice of aspect ratio r at the critical point p_c was found by Cardy [28]. The original expression, involving a hypergeometric function, can be rewritten as a simpler series expansion [289], like so

$$\Pi_h(r, p_c) = \frac{2^{7/3}\pi^2}{\sqrt{3}\Gamma(\frac{1}{3})^3}\left(e^{-\pi r/3} - \frac{4}{7}e^{-7\pi r/3} + \ldots\right). \tag{3.1}$$

This expression predicts $\Pi_h(1, p_c) = 1/2$ for the infinite (square) lattice, and this is supported by numerical results [287, 290] showing

$$\Pi_h(1, p_c) = 1/2 + b_0/L + \ldots \tag{3.2}$$

with a leading order finite size correction of order L^{-1}. In this expression, the parameter b_0 must be measured, and Monte Carlo data indicates a value of around 0.319(1) or 0.320(1) [287, 290, 296]. Ziff's original determination of b_0 [287], and of the square site percolation threshold p_c (see table 3.3), relied upon finite-size scaling relations (derived from real-space renormalisation

group based conjectures of Reynolds, Stanley and Klein [212]) that were subsequently contested by Aharony and Hovi [2], who argued that a leading order term of $L^{-0.85}$ should be expected. The matter was debated [288], before it was argued that the symmetry of the square lattice hides the $L^{-0.85}$ term [15, 116]. Recent numerical tests have upheld the conjectured scaling relations [296], and generalised tests of Cardy's formula have since been conducted by Simmons, Kleban and Ziff [137, 237]. In a similar fashion, Newman and Ziff's high precision results of 2000 [190, 191] relied upon an assumed value, of a leading order scaling exponent, that was brought into question by later Monte Carlo data [22]. Further data has since supported Newman and Ziff's value [199].

3.2.6 Schramm-Loewner Evolution

Recently, mathematical progress has been made using methods from conformal field theory [101], quantum gravity [57, 58] and, particularly, Schramm-Loewner (or stochastic Loewner) evolution [22, 293, 227, 240, 239, 101, 80]. An example of this, as detailed in [293], is the fractal dimension of critical hulls. Results from Monte Carlo calculations indicated a fractal dimension of about 1.75, leading Sapoval, Rosso and Gouyet to conjecture an exact value of 7/4 [225]. Saleur and Duplantier then made physical arguments using field theory and a Coulomb gas model to arrive at the same value [224]. The conjecture was later formally proven, by Smirnov and Werner [240], using stochastic Loewner evolution.

3.3 Review of Computational Techniques

This section provides a brief overview of some of the more common techniques used in Monte Carlo studies of percolation. The discussion is neither exhaustive nor detailed, with an emphasis upon precise methods for determining thresholds upon a lattice. No account is given, for instance, of the histogram Monte Carlo method of Hu [118]. The interested reader may wish to refer to Ziff's algorithms chapter within the Encyclopedia of Percolation [293] (most of this section follows the treatment there). For the sake of simplicity, the language used here will be that of site percolation upon a regular lattice, however, generalisations are straightforward.

3.3.1 The Simple Approach

A conceptually simple method for locating the percolation threshold, p_c, is as follows. First, make a guess p for the spanning threshold. Then, for each site on the lattice, randomly occupy that site with probability p (leaving it unoccupied with probability $1 - p$). The resulting lattice configuration is a sample from the canonical ensemble distribution (fixed occupation probability, fluctuating number of occupied sites [191, 293]). Next, determine whether or not this configuration contains a spanning cluster (of occupied sites) that reaches from one boundary of the lattice to the opposite. Repetition then yields a Monte Carlo estimate of the spanning probability (the existence probability of a spanning cluster) at site occupation probability p.

Further guesses of p enable the location of the spanning threshold by interpolation. This is typically (although not necessarily) defined as p such that the existence probability of a spanning cluster is one half [296]. Collecting spanning threshold estimates from a range of lattice sizes allows the percolation threshold (on the infinite lattice) to be found by extrapolation [296]. The necessities of trialling multiple guesses for p, and of trialling every last lattice site for every guess, make this simple approach rather slow.

3.3.2 The Binary Search Method

The binary search method is a widely used extension of the simple approach of subsection 3.3.1 [293]. Each site on the lattice is assigned a random threshold value sampled from the standard equidistribution, uniform over $(0, 1)$. Some initial value p is then chosen for the site occupation probability on the lattice. Every site with a threshold lower than p is considered occupied. Every site with a threshold greater than p is considered unoccupied. If the resulting configuration contains a spanning cluster, the lattice is considered supercritical and the value of p is adjusted downwards. If instead no spanning cluster exists, the lattice is considered subcritical and the value of p is adjusted upwards. The process is continued, stepping p up and down through a binary search pattern just as in the bisection method for finding roots of functions [7, 273]. In this manner, the value of p converges to the spanning threshold of this particular finite-sized lattice, giving six digits of precision after about twenty iterations [293].

On a square lattice of N sites, the time to take a single such sample goes

as $N \ln(N)$, a significant improvement of the simple approach [190, 293] (efficiency is increased as a result of eliminating the problem of having to guess p). Of course, many samples are taken to produce a good estimate of the spanning probability for the chosen N. As before, finite size scaling can then be used to extrapolate the spanning probability to the percolation threshold on the infinite lattice [296].

3.3.3 The Hoshen-Kopelman Algorithm

In the methods of subsections 3.3.1 and 3.3.2, it is necessary to determine the existence or otherwise of a spanning cluster. Essentially this involves establishing to which cluster each occupied site on the lattice (boundary) belongs. A brute force means of doing this is as follows. Assign to each occupied site a unique integer label (since the N lattice sites, occupied or not, form a countable indexed set, this label may as well be the site's index number). For each occupied site in sequence, examine each of its neighbours in turn to see if they are also occupied. If so, relabel every lattice site that is currently labelled with the larger of the two labels (of the site and its neighbour) with the smaller of the two. After consideration of all N sites, the end result is that all sites belonging to a common cluster will share a common label unique to that cluster. It is then a simple matter to determine if any of the clusters span by comparing the labels of sites along opposite boundaries. The drawback to this method is that each relabelling requires a second pass of all sites from the first up to the current. This makes the method rather slow, running in a time as large as order N^2.

The Hoshen-Kopelman algorithm is a single-pass multiple-labeling technique that eliminates the time consuming multiple-passing problem of the above single-labeling method [115]. Like many algorithms in statistical physics, variants were already known to the computer science field [293] (in this case, as the union/find algorithm). Among the many variants of the Hoshen-Kopelman algorithm is that of Stauffer and Aharony [246]. A collection of these has been reviewed by Martín-Herrero [173]. The key idea is using a labels-of-labels system. An illustrative implementation is as follows.

The N lattice sites are indexed $\{s_i\}_{i=1}^{N}$, and each site s_i is associated with a label ℓ_i, initialised such that $\ell_i = i$. Labels may be either proper or improper. To determine the proper label of site s_i, first set $n = 0$ and $\ell_n = i$. Then, while $\ell_{\ell_n} \neq \ell_n$ set ℓ_{n+1} equal to ℓ_{ℓ_n}, increment n by one and

continue. Eventually the proper label, ℓ_n such that $\ell_n = n$, for site s_i will be found. Intermediate labels, ℓ_n such that $\ell_n \neq n$, of this search are denoted as improper.

An improvement is possible by implementing this proper label find algorithm in such a way that all improper labels passed over during the search are also reassigned to the value of the proper label (see subsection 2.4.1). This technique is known as path compression, and makes future searches faster since there will be fewer improper labels between the site and its proper label [43, 111, 257]. When two occupied sites are found to be adjacent, their proper labels are determined and compared. If the proper labels are different, the larger valued proper label is made into an improper label by reassigning it with the value of the smaller proper label (this is a weighted union algorithm). This being done, the two adjacent sites both share the same proper label and hence are identified as members of the same cluster. Performance of the algorithm will vary with the nature of the find and union methods. In its most efficient form (weighted union/find with path compression), the algorithm can run in a time of order N [190, 191]. Once this first lattice parse is complete, a second sweep is able to relabel each site directly with its proper label if such is required [293] (of course, this requires more time).

Besides providing a fast means for identifying clusters, the Hoshen-Kopelman algorithm makes possible a powerful memory saving technique useful on large lattices or in higher dimensions [17, 246]. A d-dimensional lattice can be constructed one $(d-1)$-dimensional surface at a time [293]. The Hoshen-Kopelman algorithm allows identification of proper cluster labels on the under-construction $(d-1)$-dimensional surface from the labels of the previous $(d-1)$-dimensional surface only. Hence as the lattice is constructed, slice by slice, only two $(d-1)$-dimensional surfaces need be stored in memory at any one time. Once the proper cluster labels have been determined on the final surface, these may be compared to those that were present on the first surface The presence or absence of a matching label between surfaces indicates the presence or absence of a spanning cluster between them. The method is not of use when the sampling or analysis to be performed requires the greater detail of seeing the entire lattice at once. The memory saving property enables the simulation of much larger systems than would otherwise be possible, and so the Hoshen-Kopelman method has

been widely used [167, 208, 262, 46].

3.3.4 The Hammersley-Leath-Alexandrowicz Algorithm

This algorithm, due independently to Hammersley [25, 105], Leath [146] and Alexandrowicz [4], is often commonly abbreviated to simply the Leath algorithm. The algorithm generates only a single cluster, rather than an entire lattice configuration. This can be much faster and less memory intensive than full lattice generation and so is often useful in situations where entire multiple-cluster lattice patterns are not required. There are many variants and alternatives [246, 270], but in essence all have the form of a stochastic search through the lattice space, locating all members of a single cluster, and only the members of that cluster. Figure 3.1 was constructed by just such an algorithm.

The algorithm begins with a single occupied site that is to serve as the nucleation kernel for a stochastic aggregation process. All other sites are initially labelled as being indeterminate. The aggregation search procedure may be depth-first (stack based) or, more commonly, breadth-first (queue based) [293, 43]. Adopting a queue based search, the kernel site is placed in a queue. An occupation probability p is then chosen and the algorithm proceeds as follows.

The first occupied site in the queue is considered and each of its adjacent sites is examined in turn. If the adjacent site is found to be either occupied or unoccupied then no action is taken. If instead the adjacent site is found to be indeterminate then it is tested. The test process is a simple Monte Carlo operation resulting in the site being made either occupied, with probability p, or unoccupied (with the remaining probability of $1 - p$). If the site was made occupied then it is also added to the end of the queue. Once all of its adjacent sites have been examined in this way, the considered site is removed from the queue. This process continues until the queue is empty at which point the collection of sites is irreducible as all perimeter sites are necessarily unoccupied. The collection therefore constitutes a single cluster.

Because each site is tested at most only once, this algorithm produces clusters which have exactly the same properties as those created by the simple method of testing all lattice sites before isolating the clusters (as per subsection 3.3.1). Hence the clusters produced by the Leath algorithm are samples from the canonical ensemble at occupation probability p. Leath

algorithms are useful for studies of the invasion depth [202] in epidemic [89] or chemical [110] processes, and for examining the fractal properties of clusters [91].

3.3.5 The Hull-Walk Algorithm

The hull of a cluster is the boundary enclosing all sites within the cluster and no sites not within the cluster [273]. The boundary is not generally connected and may consist of internal and external components. Hence the hull may consist of multiple (disconnected) surfaces. A cluster's hull may be determined by taking a walk over the cluster surface [269]. However, when only the hull (or part thereof) is required it is more computationally efficient to generate it directly without forming the entire cluster. The hull-walk algorithm is similar to the Leath algorithm in that it begins with a single occupied site and grows a hull boundary from it. Rather than searching for occupied sites that are members of the cluster, a search is made for the (unoccupied) bonds lying between occupied and unoccupied sites along the hull [293]. As with the simple approach of subsection 3.3.1, a fixed site occupation probability p must be chosen in advance.

The accessible, or Grossman-Aharony, hull refers to the single hull surface external to the cluster [100]. While there is no known method for generating the Grossman-Aharony hull directly [293], a random walk can be constructed to generate the hull of a cluster in rather less time ($\sim N^{7/8}$) [294, 191] than it would take to generate the entire cluster (see figure 3.6). This provides a computational speed advantage over the Leath algorithm for situations where the reduced information obtained does not compromise the end result. An example of this is the spanning (crossing) probability method for locating the percolation threshold [287] where the cluster spans the lattice if and only if its Grossman-Aharony accessible hull spans the lattice.

The hull-walk approach was first applied by Ziff, Cummings and Stell [294], Weinrib and Trugman [272] and Grassberger [90]. Hull walk algorithms have been used, along with some assumptions about the statistical properties of hull characteristics, to measure critical exponents and the percolation threshold [286]. Deterministic (non Monte Carlo) hull-walk algorithms can be used to compute exact results on small lattices (see subsection 3.2.3).

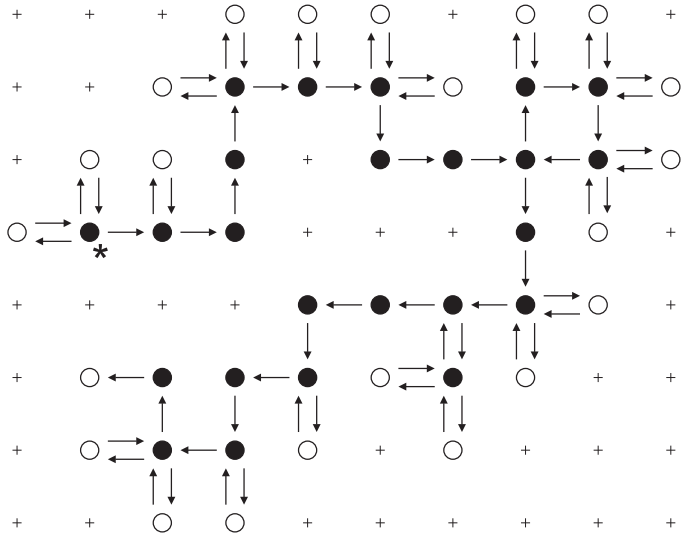

Figure 3.6: A hull generating walk in progress. Occupied sites are black, vacant sites are white, untested sites are shown as crosses. The starting point is the marked occupied site.

3.3.6 The Hull-Gradient Method

The hull-gradient method consists of a hull generating random walk algorithm that takes place upon a rectangular lattice with an imposed threshold gradient parallel to one of the sides [225, 293]. Each lattice site is given an occupation threshold as per the binary search algorithm (subsection 3.3.2), but here the thresholds are not assigned at random. Instead each site's threshold is determined by its position along the gradient axis. Without loss of generality, say that the gradient is along the lattice y axis. Further assume that this lattice side has L sites along it, in evenly spaced positions from $y = 1$ to $y = L$ inclusive. In the case of a linear gradient, all lattice sites of a given y position will be assigned a linearly interpolated threshold of $p = (y - 1)/(L - 1)$.

The random walk starts from an arbitrary site somewhere along the gradient side. As with the standard hull-walk algorithm (subsection 3.3.5), adjacent sites are tested for occupancy in order to establish the cluster perimeter. However, in the case of a gradient, sites do not all have the same occupation probability (as given above) and this provides a negative feedback mechanism to guide the walk. Figure 3.7 shows a hull-gradient

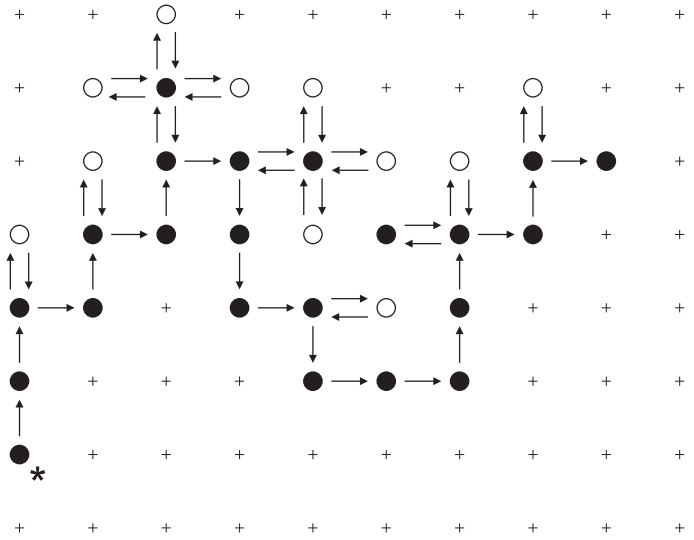

Figure 3.7: A hull generating walk on a gradient. Occupied sites are black, vacant sites are white, untested sites are shown as crosses. The starting point is the marked occupied site.

walk in progress. Sapoval, Rosso and Gouyet [225] have shown that the walk samples sites having thresholds p distributed within a well defined neighbourhood ($\sim L^{3/7}$) of the percolation threshold p_c. Hence the walk can be used to efficiently estimate this quantity [218]. Furthermore, the walk tends to loop backward (progress in the negative x direction) only for small distances. A consequence of this is that the walk progresses more or less parallel to the x axis and can be run indefinitely upon a semi-infinite lattice within finite computer memory. The hull merely delineates the lower extent of an infinite cluster connected to the upper lattice boundary side. As with the standard hull-walk, neither the entire lattice configuration nor an entire spanning cluster is generated, and this limits the algorithm's applicability. Even so, this method has provided many high precision threshold estimates on various topologies [297, 300, 249, 206], and these have been valuable in testing conjectured values.

3.3.7 The Newman-Ziff Method

The method of Newman and Ziff [190, 191, 293, 88] consists of an algorithm for populating a lattice in the microcanonical ensemble (exact number of oc-

cupied sites, inexact effective occupation probability), and a transformation to map observable quantities into the canonical ensemble (fixed occupation probability, average number of occupied sites, as per subsection 3.3.1) in order that certain data analysis techniques may be used [210, 211, 212, 287, 191, 296].

The algorithm begins with a lattice of unoccupied sites and proceeds to occupy one randomly chosen site at a time, recalculating cluster labels at each Monte Carlo step. As noted by Ziff [293], this approach had been suggested earlier by Gould and Tobochnik [87], and is equivalent to the time-dependent percolation of de Freitas, Lucena and Roux [73, 72]. It is also similar in concept to Machta, Choi, Lucke, Schweizer and Chayes' invaded cluster algorithm for the Potts model [164, 163]. However, the Newman-Ziff variant of this algorithm is a much more computationally efficient implementation. As shown in figure 3.8, the Newman-Ziff algorithm uses site tree data structures for rapid merging of clusters. Again, this is similar to the structure used by de Freitas, Lucena and Roux [73, 72]. Although they did not start out that way, the algorithms of section 2.4 have also come to use tree structures (section 2.3) as a seeming requirement of optimal computational efficiency. The trees act, essentially, as Hoshen-Kopelman label-label tables [293, 152, 115] (see subsection 3.3.3). Whenever a site's proper cluster label (see subsection 3.3.3) needs to be found, path compression [111] is applied to all improper labels encountered along the way (see also subsection 2.4.1). This find with path compression algorithm [43] has the effect of reducing future proper label search times. In addition, whenever clusters are merged, the smaller tree structure is made a subtree of the larger tree structure (this is known as a weighted union algorithm [43]) with the result that proper label search times tend to a constant for large lattices [257]. The net result of all this is that the Newman-Ziff algorithm is a very, quite possibly optimally, efficient algorithm for randomly populating a lattice. If n is the number of occupied sites on an N-site lattice (in the microcanonical ensemble), then the algorithm is able to sample one lattice configuration at each possible n in a total time of order N [190, 191] (note that these samples are, of course, correlated).

If an observable quantity Q is being sampled by the algorithm, then the result of repeated trials will be a Monte Carlo estimate for Q_n in the microcanonical ensemble. As noted, many data reduction methods, particularly

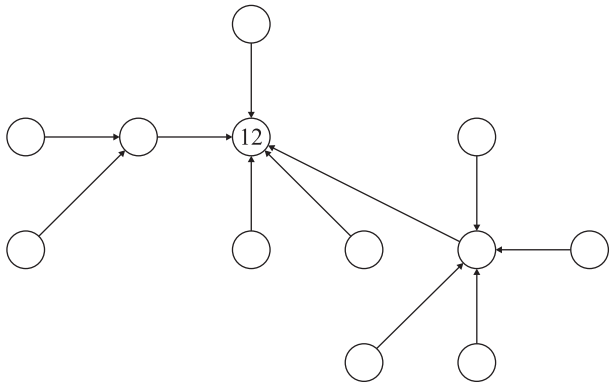

Figure 3.8: Data structure (n-ary tree) of the Newman-Ziff algorithm. Due to path compression, all sites tend to become children of the root site, upon which the total number of sites in the structure is stored. Bonds between adjacent occupied sites are not shown.

finite size scaling analysis for locating the critical point [210, 211, 212, 287, 191, 296], rely upon having access to samples from the canonical ensemble. Since the probability, π, of finding n sites occupied on a lattice of N total sites, in the canonical ensemble at occupation probability p, is given by the binomial distribution

$$\pi(n,p) = \binom{N}{n} p^n (1-p)^{N-n}, \tag{3.3}$$

it is possible to transform microcanonical ensemble data, Q_n, into canonical ensemble data, $Q(p)$, by the convolution

$$Q(p) = \sum_{n=0}^{N} \binom{N}{n} p^n (1-p)^{N-n} Q_n \tag{3.4}$$

[190, 191]. It is not possible to apply this transformation in reverse, since canonical data at any given p will contain an inextricable mixture of microcanonical n values. Any given microcanonical configuration of n occupied sites will be consistent, with some non-zero probability measure, with any canonical $p \in (0,1)$. Any canonical configuration at some given p will have non-zero probability measure for only one microcanonical n, and exactly which n is unknown.

A commonly used method for estimating the square site percolation

threshold relies upon the spanning (or crossing) function $R_L(p)$ for the probability that a single cluster exists that stretches all the way from the left boundary to the right boundary of an $L \times L$ lattice in the canonical ensemble where each site is occupied with probability p [296]. This function can be determined, by the convolution of equation (3.4), from the spanning function $R_{L,n}$ for the probability that a single cluster exists which stretches all the way from the left boundary to the right boundary of an $L \times L$ lattice in the microcanonical ensemble where exactly n sites are occupied [190, 191]. The most precise estimate of the square site percolation threshold appearing in table 3.3, was obtained by looking at wrapping clusters [295, 138, 203] on toroidal geometry, with this method [190, 191]. As noted in section 3.2, the data analysis for this estimate relied on a scaling relation that was subsequently queried but, in the end, appears to be correct [22, 199].

3.4 Techniques Utilising Edge Removal

The algorithms of chapter 2 convey the ability to efficiently deoccupy sites on the lattice (or even to remove them). This ability has several Monte Carlo sampling applications, as described below.

3.4.1 Application to the Binary Search Method

An obvious application is to the binary search method of subsection 3.3.2. If a list is made of lattice sites sorted by order of threshold then it becomes a simple matter to identify which sites are made occupied or unoccupied at each step of the binary search in p. When sites are being occupied there will be no advantage to the new algorithms over the Hoshen-Kopelman method for identifying the resulting clusters. However, when sites are being deoccupied, the clump approach provides a fast alternative to establishing the resulting clusters from scratch.

3.4.2 Sampling from the Canonical Ensemble

To sample configurations from the canonical ensemble at some fixed site occupation probability p is straightforward. The lattice is initialised, in the manner of subsection 3.3.1, so that every site is occupied with probability p (an independent Monte Carlo trial being conducted for each site).

Thereafter, sites are chosen at random, and are then randomly made either occupied (with probability p) or unoccupied (with the remaining probability $1-p$). No consideration is made of the site's prior state (nor should it be). The algorithms of chapter 2 can be used to rapidly recalculate the clusters about the site (at this point, reference to the prior state of the site will be useful to reduce computing time). The result is a new lattice configuration, differing from the prior configuration with probability $2p(1-p)$. Continuing to randomly reassign the occupancy status of randomly chosen sites, uniformly samples configurations from the canonical distribution for occupation probability p. As with the Newman-Ziff method (subsection 3.3.7), the sequence of configurations sampled will be self correlated in time.

To be valid and unbiased, this method must satisfy both the accessibility criterion and the invariance condition (as does any Monte Carlo process for an equilibrium ensemble) [81, 109, 248, 38]. The proof of this is simple. Consider a lattice of N sites, and also two arbitrary configurations, C_a and C_b, of that lattice. The accessibility criterion is obviously satisfied, since the probability of evolving the system from C_a into C_b is clearly non-zero, given N Monte Carlo steps worth of time. Now, let P_ab be the probability of transforming configuration C_a into configuration C_b with a single Monte Carlo step. Similarly, let P_ba be the probability of the inverse transformation. The symmetric construction of this Monte Carlo process implies $P_\mathrm{ab} = 0$ if and only if $P_\mathrm{ba} = 0$. If P_ab is non-zero, then C_a and C_b differ only by the occupancy of a single site. Without loss of generality, assume that C_a is the configuration with the lesser number of occupied sites. Furthermore, take the number of occupied sites in C_a to be n. Hence the probability of sampling configuration C_a from the canonical ensemble of occupation probability p is given by $\pi_\mathrm{a} = p^n(1-p)^{N-n}$, and the probability of sampling configuration C_b is $\pi_\mathrm{b} = p^{n+1}(1-p)^{N-n-1}$. To transform C_a into C_b on a single Monte Carlo step requires choosing the one unoccupied site in C_a that is occupied in C_b (with probability $1/N$), and then setting this site to be occupied (with probability p). Similarly, to transform C_b into C_a on a single Monte Carlo step requires choosing the one occupied site in C_b that is unoccupied in C_a (with probability $1/N$), and then setting this site to be unoccupied (with probability $1-p$). Hence $P_\mathrm{ab} = p/N$ and $P_\mathrm{ba} = (1-p)/N$.

It immediately follows that the detailed balance condition

$$\pi_{\mathrm{a}} P_{\mathrm{ab}} = \pi_{\mathrm{b}} P_{\mathrm{ba}} \tag{3.5}$$

is satisfied between all configurations C_{a} and C_{b}. Detailed balance implies that the invariance condition

$$\pi_{\mathrm{a}} = \sum_{\text{all b}} \pi_{\mathrm{b}} P_{\mathrm{ba}} \tag{3.6}$$

is satisfied for all possible lattice configurations, and so the proof is complete.

3.4.3 Sampling from the Microcanonical Ensemble

Samples from the microcanonical ensemble are obtained at a fixed number of occupied sites n, rather than at a fixed site occupation probability p. The Newman-Ziff method [190, 191, 293, 88], described briefly in subsection 3.3.7, consists of an efficient algorithm for generating lattice configurations in the microcanonical ensemble, and also a means of converting sample data to the canonical ensemble for analysis. That method, and also the similar methods of Gould and Tobochnik [87], and Machta, Choi, Lucke, Schweizer and Chayes [164, 163], share a common feature in that they all begin with an empty lattice and proceed to occupy individual sites (or bonds), one by one, until some stopping condition is met (typically this is the formation of a spanning cluster). The algorithms of chapter 2 allow a generalisation, from the starting and stopping conditions of the above methods, to a pair of turning conditions between which the occupation n can be swept back and forth indefinitely (as shown in figure 3.9). Provided that the turning conditions are two fixed values of n, say $n_{\min}$ and $n_{\max}$, the method takes unbiased samples from the microcanonical distribution, indistinguishable (within the domain of sweeping) from those obtained by the Newman-Ziff method (for instance, see figure 3.10). Naturally, these samples may also be transformed to the canonical ensemble by the convolution of equation (3.4). This sweeping method has the advantage that sampling can be concentrated within a domain of interest, the critical region perhaps, rather than having to repeatedly approach that domain from an initially unoccupied lattice.

That this method samples lattice configurations (from within the domain of sweeping) without bias in the microcanonical ensemble is largely

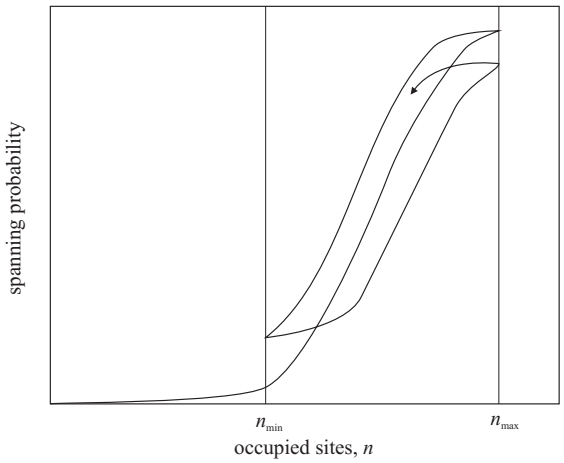

Figure 3.9: Rough indication of a typical trajectory averaged from a small number of synchronised sweeping methods operating in parallel between two fixed occupancy levels, $n_{\min}$ and $n_{\max}$, encompassing the critical region.

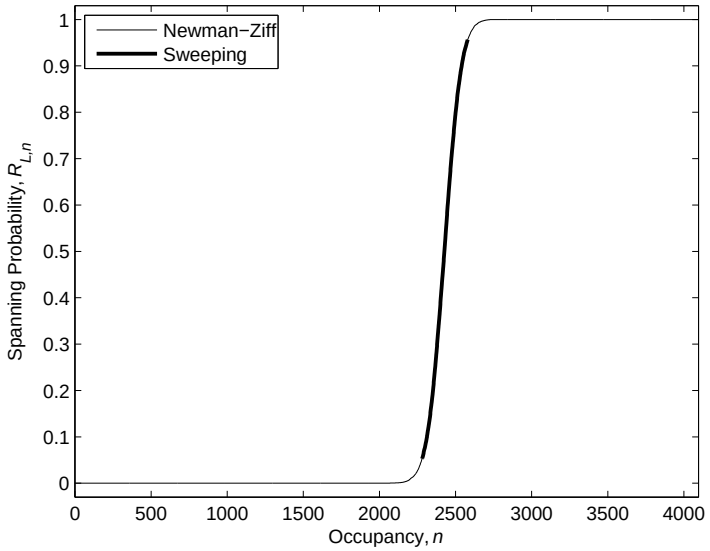

Figure 3.10: Results from the bidirectional sweeping method (subsection 3.4.3) are statstically indistinguishable from those of those of the unidirectional Newman-Ziff method (subsection 3.3.7) within the domain of sweeping ($L = 64$ square site lattice).

intuitive. A formal proof of this follows from accessibility and invariance [81]. The accessibility criterion is obviously satisfied for configurations of between $n_{\min}$ and $n_{\max}$ (inclusive) occupied sites, since the probability of evolving the system from one arbitrary configuration (satisfying this requirement) into another arbitrary configuration (also satisfying this requirement), within $2n_{\max}$ Monte Carlo steps worth of time, clearly has to be non-zero. Configurations of less than $n_{\min}$, or more than $n_{\max}$, occupied sites are not accessible (by design) and so the method is ergodic. Prior to beginning its sweeping and sampling phase, the method takes a lattice with no occupied sites and occupies one randomly selected site at a time until some predetermined number, n such that $n_{\min} \leq n \leq n_{\max}$, of sites are occupied. By construction, all possible configurations at this n are equally likely, and so the probability distribution is uniform with all configurations having an equal measure π_n determined by the inverse combination;

$$\pi_n = \binom{N}{n}^{-1} = \frac{n!(N-n)!}{N!}. \tag{3.7}$$

Now, for any given configuration C_{a} of exactly $n+1$ occupied sites, there exists precisely $n+1$ configurations of n occupied sites (and no others) that can be transformed into this $n+1$ occupied site configuration by the occupation of a single unoccupied site. In each of these $n+1$ configurations, the probability of randomly choosing the correct site to occupy is $1/(N-n)$. Hence the probability measure π_{a} of configuration C_{a} during the site occupation sweep is given by

$$\begin{aligned} \pi_{\mathrm{a}} &= \sum_{\text{all b}} \pi_{\mathrm{b}} P_{\mathrm{ba}} \\ &= \frac{n+1}{N-n}\pi_n \\ &= \pi_{n+1} \end{aligned} \tag{3.8}$$

satisfying both the invariance condition of equation (3.6) and the uniform probability condition of equation (3.7), as required. Similarly, for any given configuration C_{b} of exactly $n-1$ occupied sites, there exists precisely $N-n+1$ configurations of n occupied sites (and no others) that can be transformed into this $n-1$ occupied site configuration by the deoccupation of a single occupied site. In each of these $N-n+1$ configurations, the probability of

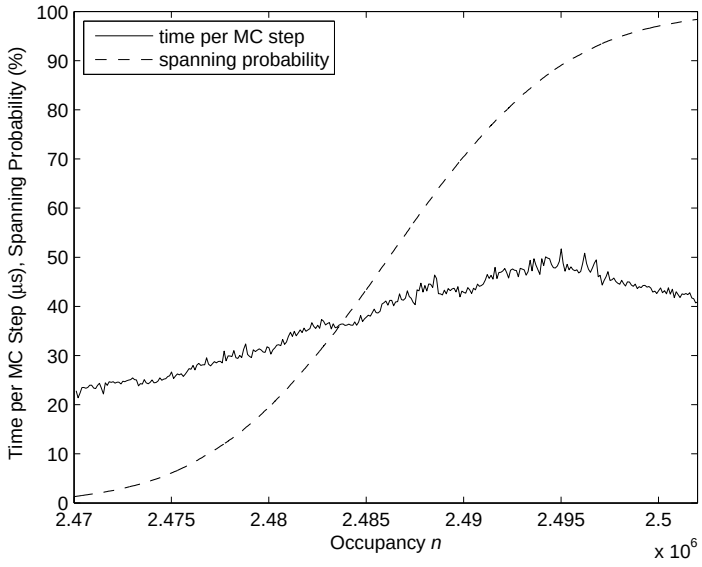

Figure 3.11: Performance of the microcanonical domain sweeping algorithm across the transition region of the $L = 2048$ square site lattice. The degree of slowing is not severe, and the point of lowest performance does not cooincide with the critical point.

randomly choosing the correct site to deoccupy is $1/n$. Hence the probability measure π_{b} of configuration C_{b} during the site deoccupation sweep is given by

$$\begin{aligned} \pi_{\mathrm{b}} &= \sum_{\text{all a}} \pi_{\mathrm{a}} P_{\mathrm{ab}} \\ &= \frac{N-n+1}{n} \pi_n \\ &= \pi_{n-1} \end{aligned} \tag{3.9}$$

satisfying both the invariance condition of equation (3.6) and the uniform probability condition of equation (3.7), as required. Hence the conditional probability of sampling a specified lattice configuration, given that the requisite number n of sites are occupied, is at all times given by the uniform probability distribution π_n of equation (3.7), and thus the proof is complete.

A drawback of this sweeping method is that, unlike the unidirectional methods it generalises, it suffers from a form of critical slowing down. This is

not severe, as shown in figure 3.11, and appears to be related to the average cluster complexity. Possibly a quantitative link could be made to the fractal dimension of cluster backbones [41, 113, 92, 213, 185, 47] or to two point correlation functions of cluster membership [60, 126, 137, 237].

In the microcanonical ensemble, the width of the critical region in two dimensions (in terms of occupied sites) increases with lattice size as $N^{5/8}$ [191] (note that in the canonical ensemble, the width of the critical region is defined in terms of the occupation probability p, and so goes as $N^{-3/8}$). Actual values for hypercubic lattices of two, three and four dimensions are given in table 3.5, and are shown graphically in figure 3.12. Although the scaling exponents in three and four dimensions should differ slightly from the two dimensional exponent [191, 88], this is not apparent from the (low quality) data. Hardware and implementation dependent run time data for this method is shown in figure 3.13. Observed run times increase proportional to N, but vary substantially with the implementation, the compiler and the hardware. They were seen to be particularly sensitive to the amount of cache, which is understandable given that the algorithms are memory intensive. This suggests that the mean computation time per Monte Carlo step increases as $N^{3/8}$ for this algorithm.

If samples from the microcanonical ensemble are to be converted to canonical ensemble data prior to analysis, then samples are not required from the entire domain of $0 \leq n \leq N$, or even from the entire critical region, but only from the narrower convolution domain (see subsection 3.3.7). The associated binomial coefficients of equation (3.3) can be approximated by a Gaussian

$$\binom{N}{n} p^n (1-p)^{N-n} \approx \frac{1}{\sqrt{2\pi N p(1-p)}} \exp\left(-\frac{(n-Np)^2}{2Np(1-p)}\right) \tag{3.10}$$

[273], so that the convolution distribution function can be said to have an approximate standard deviation of

$$\sigma_c = \sqrt{Np(1-p)}. \tag{3.11}$$

Note that the Gaussian approximation to the binomial may not be adequate for high precision calculations, particularly on smaller lattices [191]. Parameters for this distribution near the critical point on two dimensional square

D	L	N	$n_{0.1}$	$n_{0.9}$	W
2	32	1024	540	670	130
	48	2304	1260	1470	210
	64	4096	2280	2580	300
	96	9216	5210	5710	500
	128	16384	9350	10070	720
	192	36864	21250	22450	1200
	256	65536	38000	39700	1700
	384	147456	85950	88850	2900
	512	262144	153300	157500	4200
	768	589824	346000	353000	7000
	1024	1048576	616000	627000	11000
	1536	2359296	1390000	1407000	17000
	2048	4194304	2474000	2498000	24000
3	16	4096	1100	1500	400
	24	13824	4000	4800	800
	32	32768	9800	11000	1200
	48	110592	33200	36400	3200
	64	262144	80000	85000	5000
	96	884736	271500	283000	11500
	128	2097152	646000	664000	18000
4	8	4096	650	1050	400
	12	20736	3600	4800	1200
	16	65536	11800	14500	2700
	24	331776	62000	70000	8000
	32	1048576	200000	216000	16000

Table 3.5: Approximate upper and lower boundaries of the critical region for hypercubic lattices of $N = L^D$ sites in two to four dimensions D. The upper and lower boundaries are defined as the number of occupied sites, n, for which the spanning probability $R_{L,n}$ is equal to 0.9 ($n_{0.9}$) and 0.1 ($n_{0.1}$) respectively. The width of the critical region is defined as $W = n_{0.9} - n_{0.1}$.

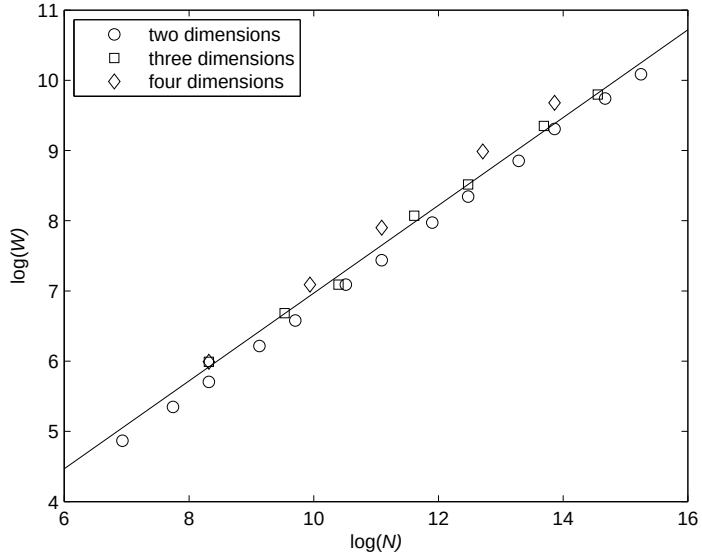

Figure 3.12: In the microcanoncial ensemble, the width W (in sites) of the critical region, scales as $W \propto N^{5/8}$ for simple hypercubic lattices of N total sites. Data is from table 3.5 (low precision values).

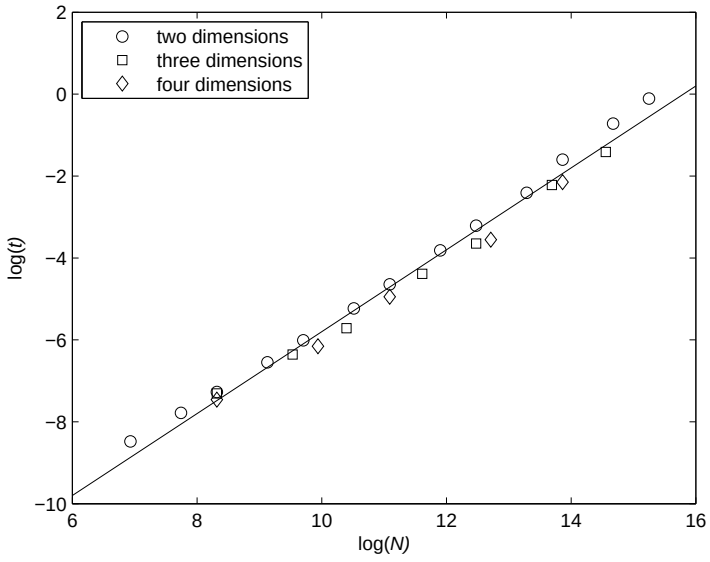

Figure 3.13: Performance of the algorithms while sweeping the critical region between fixed turning points, $n_{0.1}$ and $n_{0.9}$, as specified in table 3.5. The observed run time per sweep goes approximately as $t \propto N$.

L	$\bar{n}_c$	σ_c	$\bar{n}_c - 3\sigma_c$	$\bar{n}_c + 3\sigma_c$
32	607	16	560	654
64	2428	31	2334	2522
128	9712	63	9523	9900
256	38846	126	38469	39224
512	155385	252	154630	156140
1024	621539	503	620030	623049
2048	2486158	1006	2483139	2489176
4096	9944631	2012	9938594	9950668

Table 3.6: Centre point, $\bar{n}_c = pL^2$, and approximate standard deviation, $\sigma_c = L\sqrt{p(1-p)}$, of the convolution region (binomial distribution), in two dimensions at site occupation probability $p = 0.5927462$ (see subsection 3.3.7 and equation (3.10)).

site lattices are listed in table 3.6, and the $N^{1/2}$ increase of the convolution domain is shown in figure 3.14. Wall clock run time data (figure 3.15) increases as $N^{7/8}$ over these same lattices. Hence observations over both the critical and convolution regions suggest that, near the critical point, the run time per Monte Carlo step increases as $N^{3/8}$ for this method (just as for sweeping the critical region).

3.4.4 Self-Organised Critical Sampling

By redefining the two turning conditions of the bidirectional microcanonical ensemble sampling method (subsection 3.4.3) as a change in the order parameter, rather than as two fixed occupation levels, it becomes possible to operate the algorithms of chapter 2 in a self-organised critical fashion [93, 9, 267, 8] analogous to the method of Tomita and Okabe [263]. For example the occupancy could be increased until the lattice goes supercritical (*i.e.* until a spanning cluster exists), and then the occupancy could be decreased until the lattice returns to a subcritical state (with this cycle repeated indefinitely). This procedure does not produce unbiased samples from the microcanonical ensemble since not all configurations at a given occupation level n will be accessible. However, the bias is small enough that the results can be used to locate fixed turning points, encompassing the critical region, near enough for subsequent use by the method of subsection 3.4.3. This allows the algorithms of chapter 2 to be used in percolation stud-

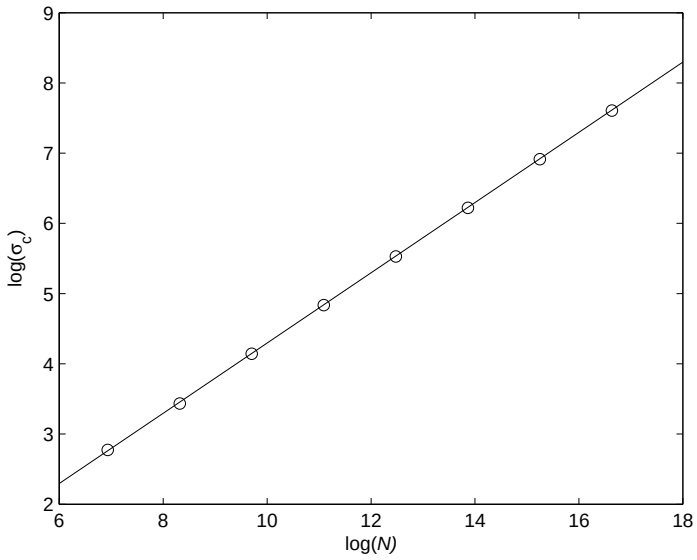

Figure 3.14: At $p \approx p_c$, the width of the convolution region, σ_c (see table 3.6), scales as $\sigma_c \propto N^{1/2}$.

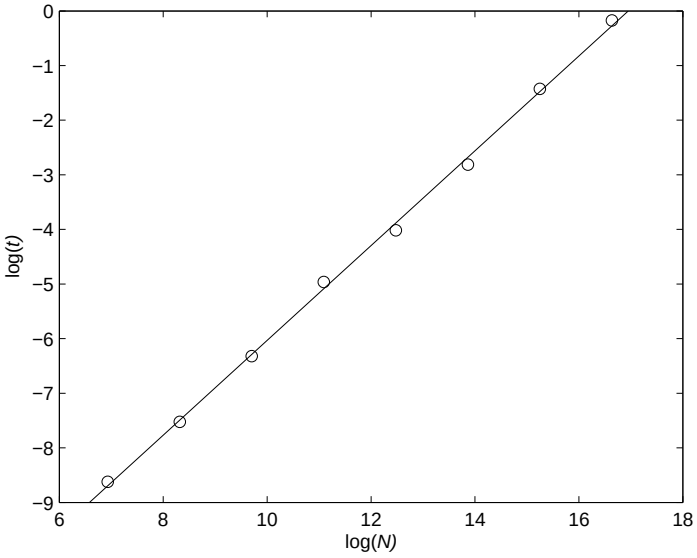

Figure 3.15: The wall clock run time, t, for a back and forth sweep of the convolution region, is observed to go as $t \propto N^{0.87}$ on square lattices of N sites ($0.87 \approx 7/8$).

ies without any prior information about the system (see table 3.7 and figure 3.16). A feature of this self-organised sweeping, is that the algorithms take the lattice on a random walk through configuration space, traversing only configurations that are right at the phase change threshold, plus or minus one occupied site from making the transition. An interesting alternative interpretation is that the algorithms hold the system at the critical point while its apparent location moves around due to finite size effects.

Original performance testing of the sweeping method was conducted on sixty-four bit hardware and with self-organised critical turning conditions. It was found that the wall-clock time required to make a sampling sweep scaled as $N^{0.95(1)}$ in two-dimensions and as $N^{0.85(5)}$ in three through seven dimensions. In comparison, the Newman-Ziff algorithm execution time scaled as $N^{1.1(5)}$ on the same hardware. However, this favourable scaling observed for the sweeping method largely vanished on the thirty-two bit Blue Gene platform used later. In practice, actual performance will depend strongly upon the hardware platform, the details of algorithm implementation, the compiler and operating system.

3.5 Preliminary Threshold Estimate

As a demonstration of the algorithms' (of chapter 2) microcanonical ensemble sampling capabilities (subsection 3.4.3), a quick estimate of the square site percolation threshold was made. This used both an approximation and a previously determined parameter, as detailed below.

3.5.1 Sampling

Lattice configurations were generated by the Monte Carlo method of subsection 3.4.3. Two different pseudorandom number generators were used. All initial trials, and some later runs, used a two-tap lagged Fibonacci generator, while the majority of runs used the Mersenne twister generator. In both cases, some runs used a single generator to select lattice sites by their index, while others used a pair of generators of the same type to select lattice sites from their coordinates. The latter approach is slower but eliminates lower order bits from each pseudorandom output word thereby improving (in principle) reliability (see subsection 3.6.1).

L	S1	S2	S3
16	1280(60)	1350(50)	1400(50)
24	4310(130)	4460(100)	4570(100)
32	10230(220)	10470(180)	10670(180)
48	34500(500)	35000(400)	35450(370)
64	81600(800)	82640(670)	83300(640)
96	275800(1800)	277600(1400)	279600(1400)
128	654000(3000)	657000(2500)	660000(2300)
192	2206000(6200)	2212000(5000)	2218000(5000)
256	5224000(10300)	5240000(9000)	5246000(9000)

Table 3.7: Sampling regions, given as a centrepoint and standard deviation of the sampling density, of the self-organised critical method on simple cubic lattices of $N = L \times L \times L$ sites. The turning points are specified as the existence or otherwise of a cluster spanning any one (S1), any two (S2), or all three (S3) lattice dimensions respectively.

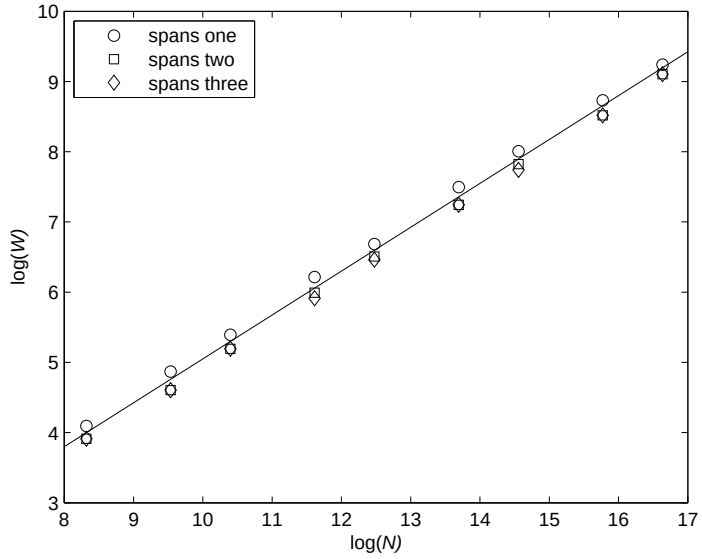

Figure 3.16: The width W (in sites) of the self-organised critical sampling region (see table 3.7) goes as $W \propto N^{5/8}$ on simple cubic lattices of N sites, just as for the critical region (see figure 3.12). It does not appear to matter whether the SOC turning condition is taken to be the presence or absence of a cluster spanning one, two or three dimensions of the lattice.

The majority of this exercise was conducted upon about a dozen GNU/ linux boxes of assorted vintage. These machines occupied space in an undergraduate laboratory and, as such, were subject to frequent and irregular rebooting without warning. This imposed a requirement of short run times with frequent saving of data to file. Combined with very severe storage space limitations, this dictated a narrow domain of sweeping, $2485700 \leq n \leq 2486700$, on the chosen $L = 2048$ lattice (this being the largest model that would comfortably fit into all of the machines). Although much data was also obtained (later) with an IBM p575 high performance computer system (acquired by the university toward the end of this exercise), consistency requirements led to the narrow sweeping domain being retained throughout.

Because this sweeping domain is narrower than the convolution region (see table 3.6), the transformation of equation (3.4) cannot be reliably applied to the data. This would be problematic, since the data is sampled from the microcanonical ensemble and most analysis requires canonical ensemble data, except that, for large L, the two distributions are approximately equal in the vicinity of the critical point. Hence, for some observable Q,

$$Q(p = n/N) \approx Q_n \tag{3.12}$$

provided that $p \approx p_c$, and, interpolating for $pN \notin \mathbb{Z}$,

$$Q(p) \approx Q_{\lfloor pN \rfloor} + (pN - \lfloor pN \rfloor) \left(Q_{\lceil pN \rceil} - Q_{\lfloor pN \rfloor} \right) \tag{3.13}$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x, and $\lceil x \rceil$ denotes the smallest integer not less than x (*i.e.* the rounding of x to the nearest integer in the downwards and upwards directions respectively).

The raw observable in this exercise was $s_{d,n}$, the number of occasions in which the lattice was found to have a total of n occupied sites and at least one cluster spanning right across the lattice in d or more dimensions (unspecified directions). Clearly, $s_{0,n}$ is the total number of observations made in which n sites were occupied (and, by construction, is independent of n within the domain of sampling). The existence probability for a cluster spanning the lattice in some specified direction within the microcanonical ensemble, $R_{L,n}$, can then be determined and is approximately equal to the

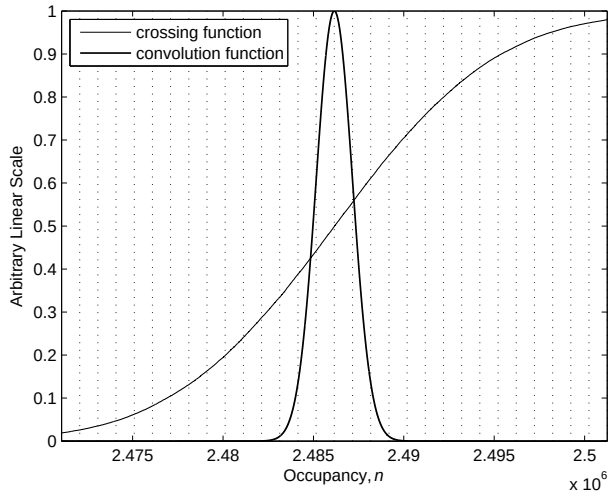

Figure 3.17: For the square site lattice of $L = 2048$, the spanning (crossing) probability function $R_{L,n}$ is approximately linear over the narrower convolution domain. Dotted vertical lines indicate one standard deviation (of the convolution distribution) intervals.

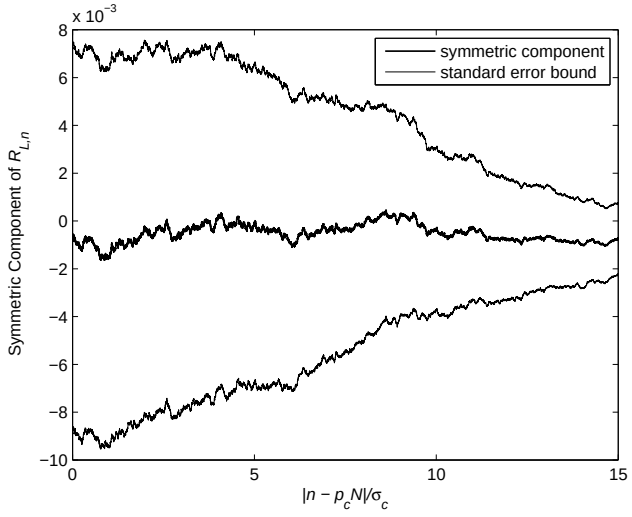

Figure 3.18: The microcanonical ensemble spanning probability curve $R_{L,n}$ appears (within uncertainties) to be an odd function about the point $R_{L,n} = 1/2$, out to fifteen standard deviations of the convolution distribution. To be precise, the function plotted is $(R_{L,p_cN+n'} + R_{L,p_cN-n'} - 1)/2$, $n' \in \mathbb{N}_0$, with $p_c = 0.5927462$.

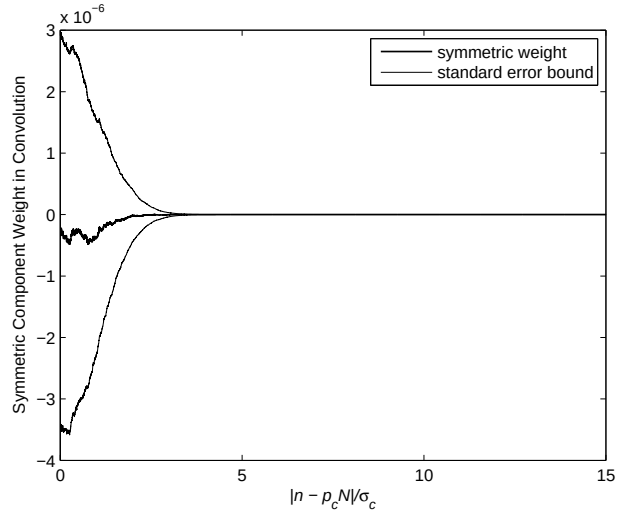

Figure 3.19: Symmetric component of $R_{L,n}$ (as in figure 3.18) multiplied by the corresponding binomial coefficient of the convolution function. This is the contribution that each term (each n) will make to the sum of equation (3.4). All points here are zero within uncertainties.

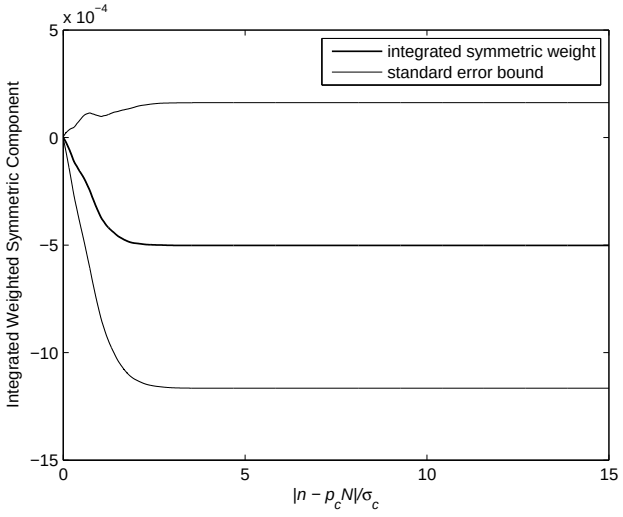

Figure 3.20: Integral of the function of figure 3.19. This is the total contribution of the even component of $R_{L,n}$ to the sum of equation 3.4, and is zero within uncertainties, suggesting that $R_L(n/N) \approx R_{L,n}$ for $p \approx p_c$.

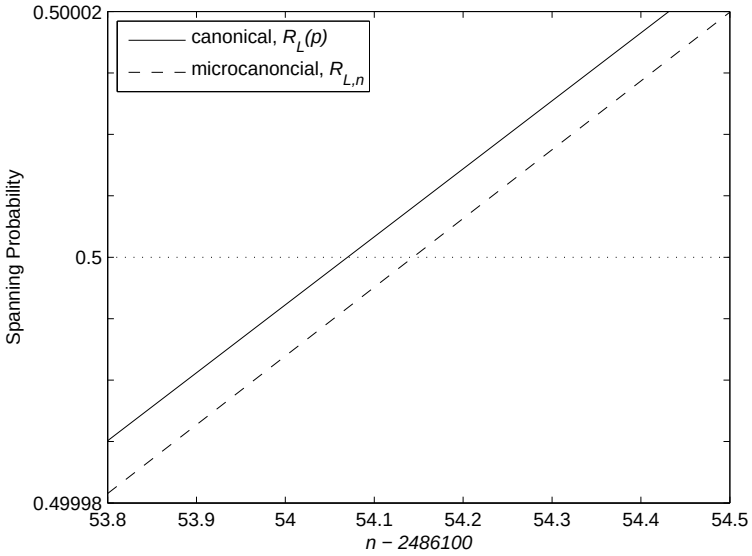

Figure 3.21: Higher quality data ($L = 2048$) shows that n/N such that $R_{L,n} = 0.5$ typically exceeds p such that $R_L(p) = 0.5$ by a little less than 2×10^{-8}. The statistical uncertainties (not shown) in p and n/N are over twice this number.

equivalent function in the canonical ensemble, $R_L(p)$, provided $p \approx p_c$.

$$R_L(p = n/N) \approx R_{L,n} = \frac{s_{1,n} + s_{2,n}}{2s_{0,n}} \tag{3.14}$$

Based on a limited amount of data from a wider sweeping domain, this approximation was believed to be adequate, with an error of no more than $O(10^{-8})$ or so. Figure 3.17 shows that the spanning probability curve $R_{L,n}$ is, at least approximately, an odd function about the centre of the convolution domain (this is so because the critical point coincides with the inflection point of $R_{L,n}$ [296]). In figure 3.18, all odd components have been removed from $R_{L,n}$, and the remainder is zero within uncertainties. Hence all non-cancelling terms of the convolution sum are zero within uncertainties (figure 3.19) and the integral of the non-cancelling terms (that is, the total difference between $R_{L,n}$ and $R_L(p)$ at $p \approx p_c$) is also zero within uncertainties. It was also found that this cancellation was not particularly sensitive to p. This data is relatively low precision however. Better quality data obtained later

(see subsection 3.6.3 and figure 3.21) showed that use of $R_{L,n}$, rather than $R_L(p)$, typically results in an overestimate of the critical point by around 2×10^{-8} on an $L = 2048$ lattice. This is somewhat less than statistical sampling uncertainties.

3.5.2 Statistical Errors

Define $p_{d,n}$, $d \in 0, 1, 2$, as the existence probability of a single cluster that crosses (spans) the lattice in exactly d dimensions (without specifying any particular direction) in the microcanonical ensemble with n sites occupied. That is, a cluster satisfying the condition of $d = 1$ must span the lattice either from the left boundary to the right boundary, or else from the top boundary to the bottom boundary, but not both. Hence, for a finite number of Monte Carlo samples (as above in subsection 3.5.1), $p_{0,n} \approx (s_{0,n} - s_{1,n})/s_{0,n}$, $p_{1,n} \approx (s_{1,n} - s_{2,n})/s_{0,n}$, and $p_{2,n} \approx s_{2,n}/s_{0,n}$. The crossing probability spanning a single, specified, direction, $R_{L,n}$, is given by

$$R_{L,n} = \frac{1}{2} p_{1,n} + p_{2,n} \tag{3.15}$$

just as in equation (3.14). For $R_{L,n} \approx 0.5$ (equivalent to $p \approx p_c$, or, equivalently, to $n \approx p_c N$, on a large lattice), Monte Carlo data shows that $p_0 \approx p_2 \approx 0.32$ and $p_1 \approx 0.34$ (dropping the implied subscripts of $n \approx p_c N$). Note that equation (3.15) implies $p_0 = p_2$ exactly at $R_L = 0.5$. Hence the standard deviation, σ_R, in a set of Monte Carlo data for R_L is given by

$$\begin{aligned} \sigma_R^2 &= \langle \left(R_L - \overline{R_L} \right)^2 \rangle \\ &= p_0 (0 - 0.5)^2 + p_1 (1/2 - 0.5)^2 + p_2 (1 - 0.5)^2 \\ &= \frac{1}{2} p_0 \end{aligned} \tag{3.16}$$

so that $\sigma_R \approx 0.4$. This a small improvement over methods that choose a specific spanning direction in advance [287], for which $\sigma_R = 0.5$.

Now, since $R_{L,n}$ is determined from some number, $s_{0,n}$, of configuration samples, the standard error in R_L is given by the standard deviation of the mean. Furthermore, samples from the sweeping method are not independent, but are correlated with autocorrelation time τ. The value of τ is dependent upon the width of the sweeping domain (more precisely, upon the ratio of the number of sites within the domain to the total number of sites

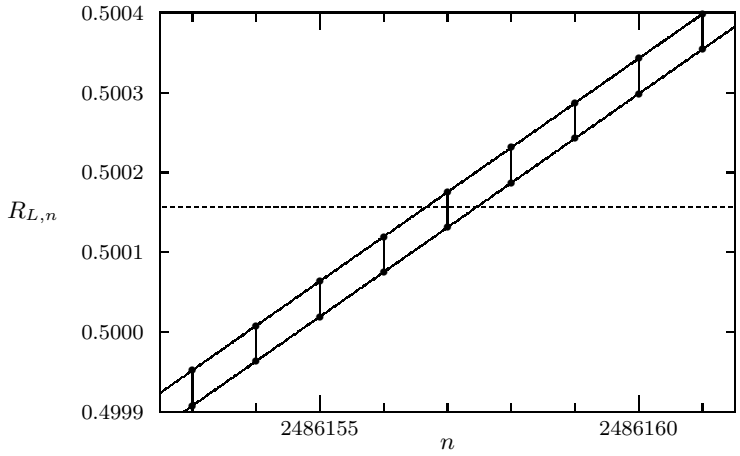

Figure 3.22: Detail of the spanning probability results in the neighbourhood of the critical point. This data suggests a threshold estimate of $p_c = 0.59274603(9)$.

on the lattice). It follows that the standard error in Monte Carlo estimates of the spanning probability function is given by

$$\sigma_{\overline{R}} = \sqrt{\frac{p_0\tau}{2s_0}} = \sqrt{\frac{s_2\tau}{2s_0^2}} \tag{3.17}$$

for this method. Very good agreement was found between this expression and statistics calculated from experimental data.

3.5.3 Results

The Monte Carlo calculations of subsection 3.5.1 return microcanonical ensemble spanning probability data $R_{L,n}$. The ensemble equivalence approximation of equation (3.14) is made (see figure 3.21), thereby allowing the percolation threshold, p_c, to be found from

$$R_L(p_c) \approx 0.5 + 0.320(1)/L = 0.50015625(50) \tag{3.18}$$

(see subsection 3.2.5) [287, 296]. Details about the neighbourhood of the point of intersection are shown in figure 3.22. Note that this figure only includes data derived from the Mersenne twister generator.

Data from the lagged Fibonacci generator produced an estimate of $p_c = 0.5927466(2)$, while data from the Mersenne twister gave $p_c = 0.59274603(9)$

[152]. In either case, it did not appear to matter whether mono or dual generators were used (see subsection 3.5.1), nor was any dependence found upon the method of seeding. The combined result from both of these generators is $p_c = 0.59274616(8)$, consistent with the high precision value, $p_c = 0.59274621(13)$, of Newman and Ziff [190, 191]. However, since the results from the two different generators are not exactly consistent, the combined results should be treated with caution. Although the Newman and Ziff result was itself obtained with a two-tap lagged Fibonacci generator [190, 191], these have performed relatively poorly in statistical tests, leading some authors to caution against their use (Ziff included) [291, 149]. The Mersenne twister is a very well regarded generator [266, 149], yet is not infallible, and the Mersenne based result is a little lower than would be comfortably expected (the results of table 3.3 would indicate a value of around $p_c = 0.5927463(1)$). Extensive testing had ruled out any bug or systematic error in the computer program, so, if the result obtained here was in error, its source would be found in equation (3.18) (unlikely [287, 296], and later ruled out in subsection 3.6.4), the approximation of equation (3.14) (the resulting error was later found to be small in subsection 3.6.3, where the estimate of $p_c = 0.59274603(9)$ was revised only slightly to $p_c = 0.5927460(1)$), or bias from the chosen random number generator (this seemed likely in light of the results obtained here). Ziff suggested that many high precision results based on different generators would help resolve the issue [285]. Such a survey is covered within section 3.6, where the discrepancy was indeed attributed to the generators. However, the best square site percolation threshold estimate obtained from the brief exercise of this current section remains $p_c = 0.59274603(9)$ [152].

3.6 High Precision Threshold Measurement

With the university's acquisition of a Blue Gene supercomputer, it became possible to save, not all but enough of, the microcanonical Monte Carlo sample data to disk for later transformation into canonical data. This was not due to the improved computing capacity in any way, but merely to the simple increase in storage space. As for the computing capacity, the ability to run a thousand parallel processes, rather than merely tens, drastically reduced the time required to obtain useful quantities of data. This in turn

Acronym	Algorithm
TT	$x_i = x_{i-418} + x_{i-1279}$
TTT	$x_i = (u_i = u_{i-471} \oplus u_{i-9689}) \oplus (v_i = v_{i-30} \oplus v_{i-127})$
SWB	$x_i = x_{i-222} - x_{i-237} - \beta_{i-1}$
QTA	$x_i = x_{i-157} \oplus x_{i-314} \oplus x_{i-471} \oplus x_{i-9689}$
QTB	$x_i = x_{i-471} \oplus x_{i-1586} \oplus x_{i-6988} \oplus x_{i-9689}$
XG	xor4096i
MT	MT19337ar
DMT	(MT19937ar,MT19937ar)

Table 3.8: Concise definitions for the primary pseudorandom number generators employed in this exercise.

meant that additional problems could be addressed within a fixed amount of time. Hence, with the new facility, the approximations of the previous section could be dispensed with and results from various pseudorandom number generators could be compared. In so doing, these two potential sources of systematic error can be removed and accurate measurements to very high precision become possible. To begin with, a suitable pseudorandom number generator must be found [153].

3.6.1 Pseudorandom Number Generators

Pseudorandom number generators are used as a source of essentially random numbers since noisy physical devices are often inconvenient or expensive, produce output sequences that are not reproducible, and, more importantly, often fail statistical tests badly [79, 169, 147]. The specific pseudorandom number generators to be considered within this exercise shall be identified by acronyms and are defined as follows (with a brief summary given in table 3.8). The symbol $\oplus$ denotes bit-wise logical exclusive-or. The symbol $\triangleright m$, with $m \in \mathbb{N}$, denotes shift m bits to the right and is equivalent to integer division by 2^m. The symbol $\{a : b\}$ denotes the set of all integers not less than a and not greater than b.

TT is the two-tap Mitchell and Moore [140] additive lagged Fibonacci generator $x_i = x_{i-418} + x_{i-1279}$. This generator has previously been used for high-precision percolation threshold measurement by Newman and Ziff [190, 191].

TTT combines the output from a pair of Kirkpatrick and Stoll [136]

two-tap generalised feedback shift-register generators [156, 258, 192], $u_i = u_{i-471} \oplus u_{i-9689}$ and $v_i = v_{i-30} \oplus v_{i-127}$, to return a single word $x_i = u_i \oplus v_i$. This is the generator most likely used for two and three dimensional percolation by Deng and Blöte [46, 21].

SWB is a Marsaglia and Zaman subtract with borrow generator, $x_i = x_{i-222} - x_{i-237} - \beta_{i-1}$, where the borrow, β_i, is equal to one if $x_{i-222} < x_{i-237} + \beta_{i-1}$, and is otherwise equal to zero [172, 170].

QTA is the quad-tap generalised feedback shift-register generator $x_i = x_{i-157} \oplus x_{i-314} \oplus x_{i-471} \oplus x_{i-9689}$, as used by Ziff and Stell [291, 287] (see [291]).

QTB is the quad-tap generalised feedback shift-register generator $x_i = x_{i-471} \oplus x_{i-1586} \oplus x_{i-6988} \oplus x_{i-9689}$. This generator has been used by Newman and Ziff, and has been found to produce threshold estimates consistent with those of the TT generator [190, 191, 296].

XG is Brent's xor4096 generator [24]. Specifically, the implementation xor4096i, from his C language xorgens304 distribution, was that used here. This generator has performed well in randomness tests conducted by L'Ecuyer and Simard [149].

MT is Matsumoto and Nishimura's MT19937 Mersenne twister generator [177]. Specifically, their MT19937ar C language distribution was the implementation used here. The MT19937 algorithm has been used for computing integrals in semi-rigorous work by Balister, Bollobás, Walters and Riordan [10, 214].

DMT is a pair of MT generators operated entirely independently of one another. The output sequence from each of these generators is decimated, with only every fourth word used. Lattice sites are then selected by means of their Cartesian coordinates, using one number from each generator. This is the scheme used previously in section 3.5. To be completely unambiguous, let L be the number of sites lying on each edge of an $L \times L$ square lattice. Every site on that lattice is typically given a unique index or label, $j \in \{0 : L^2 - 1\}$. In a microcanonical ensemble Monte Carlo calculation [190, 191, 296], such as those performed here, generator output words, x_i, are used to pseudorandomly select sites, s_j, for occupation. Now, consider a transformation $T(x, N) \equiv x \rhd (w - \log_2 N)$, where N is a positive-integer power of two. This is a distribution preserving many-to-one surjective map from the integers $x \in \{0 : 2^w - 1\}$ to the integers $T(x, N) \in \{0 : N - 1\}$.

Further consider a bijection H that maps the integers $\{0 : N-1\}$ onto site labels. With the usual choice of site labels also being the integers $\{0 : N-1\}$, H is conventionally taken to be the identity map. In the single generator systems defined above, generator output word x_i is associated with site $s_{j(x_i)}$ via $j(x_i) = H(T(x_i, L^2))$. For the DMT, a pair of generator output words, u_i and v_i (one from each of the output decimated MT generators), is mapped to site $s_{j(u_i,v_i)}$ via $j(u_i, v_i) = H(T(u_i, L) + LT(v_i, L))$. This halves the number of bits actually used from each output word (the most significant bits being those retained).

Each of these generators must be provided with a finite length sequence of initial words from which to begin calculating an infinite pseudorandom sequence. In the case of the TT generator for instance, a list of some 1279 initial words is required. These initial lists were constructed by one of four simpler generators, here denoted LCGa, LCGb, LCGm and WMx. LCGa is the linear congruential generator [148, 154], $x_i = 69069x_{i-1} + 1$, suggested by Marsaglia [168]. LCGb is a similar linear congruential generator, $x_i = 69069x_{i-1} + 1234567$, also due to Marsaglia [170]. LCGm is the modified linear congruential generator, $x_i = 1812433253(x_{i-1} \oplus (x_{i-1} \triangleright 30)) + i$, appearing in Matsumoto and Nishimura's MT19937ar distribution of their Mersenne twister algorithm. The multiplier is due to Knuth [177, 140]. WMx is the Weyl modified Marsaglia xorshift generator built into the xor4096i algorithm appearing within Brent's xorgens304 distribution [24, 171]. These initialisation generators are themselves seeded from a single word, $x_0 \in \{0 : 2^w - 1\}$. The TTT and DMT generators both require two initialisation lists, each being derived from one of these four generators, each starting with a distinct independent seed word.

3.6.2 Correlation Detection

Correlations inevitably found in the output sequence of any deterministic pseudorandom number generator will result in correlations within the spatial pattern of occupied sites upon the lattice. As noted by Compagner [39] and Gammel [79], this in turn will bias the resulting Monte Carlo estimate (in this case, of the crossing probability function). Consequently, when estimates obtained from two different generators are inconsistent, then at least one of those generators likely suffers from significant correlations in its output sequence, hence rendering it unsuitable for use at the level of

s_0	s_1	s_2	s_3
s_4	s_5	s_6	s_7
s_8	s_9	s_{10}	s_{11}
s_{12}	s_{13}	s_{14}	s_{15}

(a)

s_0	s_{11}	s_6	s_1
s_{12}	s_7	s_2	s_{13}
s_8	s_3	s_{14}	s_9
s_4	s_{15}	s_{10}	s_5

(b)

Figure 3.23: Labellings of the $L = 4$ square site lattice. (a) is the standard enumeration, (b) is one of the $16! - 1$ possible non-standard enumerations.

precision of the study. Because the true values of the spanning functions $R_{L,n}$ and $R_L(p)$ (for definitions see subsection 3.3.7), and of the percolation threshold p_c, are unknown, it will be unclear as to which of the generators is deficient.

While there is merit in performing general tests on pseudorandom number generators, it is often preferable to have an application specific test [67] such as the sensitive hull walk of Ziff [291]. Here a scheme is used that changes the relation between numerical generator output sequences and the spatial patterns of occupied lattice sites, without altering the underlying problem or topology in any way. The standard enumeration of the lattice, as shown in figure 3.23, prescribes a specific relation between patterns in generator output and in clusters of occupied sites. By adopting some other (non-standard) enumeration, as per the example in figure 3.23, some different relation is obtained. An ideal random number generator will produce results independent of the chosen enumeration [31, 141]. A pseudorandom number generator, with correlations in its output sequence, will produce results that do depend upon the enumeration. By comparing results from a common generator on two different lattice enumerations, inadequate, outcome biasing, generators may be identified. This simple application specific test does not require knowledge of the percolation threshold or spanning probability curves.

The direct approach to implementing such an enumeration is to allocate each site a set of pointers explicitly identifying its geometrical neighbours. In the non-standard enumeration of figure 3.23, for example, s_7 would have pointers to s_2, s_3, s_{11} and s_{12}. Sites are pseudorandomly selected as per normal and the Monte Carlo sampling proceeds just as for the standard

enumeration. In practice this results in a dramatic performance decrease of the simulations (more than a factor of two was found in this study). The problem is believed to be the cache prefetching of the high performance computer system used, where if, in some linear array, s_j is being accessed then the hardware assumes that s_{j+1} (being the next contiguous data element in memory) will be wanted next. The hardware then loads that element into cache in advance, but when it so happens that some other element was required instead, then that other element must be retrieved from normal memory and the performance gain usually resulting from the prefetching is lost.

An alternative method is to change the mapping, H, between scaled generator output words, $y = T(x, N)$, and site labels, j. In the standard enumeration of figure 3.23, $H(y) = y$ is the identity map. In the non-standard enumeration, $H(0) = 0$, $H(11) = 1$, $H(6) = 2$, and so on, with the general relation being $H(y) = 3y \pmod{16}$. This is analogous to the cluster label labels of Hoshen and Kopelman [115]. The non-standard enumeration is that effectively in use, while the standard enumeration is preserved in computer memory, thus avoiding performance problems.

When the hash function, H, is simple (that is, of similar algebraic complexity to the pseudorandom number generator [31, 141]), it will not so much hide correlations in the output sequence as manifest those patterns in some other way, giving rise to a different estimate for $p_{\mathrm{f}}(L)$. If, on the (systematic) standard enumeration, correlations in generator output lead to spatial correlations of occupied sites that in turn bias the estimate, then, on some other (systematic) non-standard enumeration, those same output sequence correlations will give rise to spatial correlations of a different nature that bias the estimate in some other way. This provides a simple, application specific, test for pseudorandom number generator biasing of the Monte Carlo samples. If a given generator is correlation free, then the estimates derived from it will be independent of the lattice enumeration. If instead, the generator output does suffer from correlations, then different enumerations may lead to different results. The test can be tuned, with the hash chosen so as to maximise the observed shift in the test quantity. A simple hash related to the taps or period is bound to highlight intrinsic generator shortcomings [39]. Alternatively, a hash much more complex than the generator algorithm could go some way toward hiding output sequence correlation induced bias.

Hence define two further generators, TTH and MTH, as (respectively) the exact same TT and MT generators defined previously, but with a somewhat arbitrary non-trivial mapping $H(y) = 947y \pmod N$ between integers $y \in \{0 : N-1\}$ and site labels $j \in \{0 : N-1\}$. On a lattice of $L = 2048$, this hash is equivalent to a systematic non-standard enumeration where the rightward and downward neighbours of site s_j are (when they exist) $s_{j+3171195 \pmod N}$ and $s_{j+1824768 \pmod N}$ respectively.

3.6.3 Finding a Suitable Generator

On the square site lattice there is no convenient analytical result against which statistical Monte Carlo estimates from various pseudorandom number generators may be tested. By using several different generators to sample microcanonical crossing probability functions, $R_{L,n}$, the generators may be tested against each other. Here the lattice size shall be fixed at $L = 2048$ and the algorithms of chapter 2 will be used (in the manner of subsection 3.4.3) to sample $R_{2048,n}$ over the domain $n \in \{2474000 : 2498300\}$. This domain encompasses the critical region and is large enough that the truncation induced error in the canonical probability curve, $R_L(p)$, as determined by the binomial convolution of equation (3.4), is completely negligible at less than 10^{-15}.

The canonical crossing probability curve, can be used to identify a site occupation probability, $p_{\mathrm{f}}(L)$, defined such that

$$R_L(p_{\mathrm{f}}(L)) = 1/2 + k/L, \tag{3.19}$$

where $k = 0.320(1)$, as determined by Ziff and Newman [296] (their parameter b_0, see equation (3.2)). This definition of $p_{\mathrm{f}}(L)$ is chosen so as to provide a fixed reference for the comparison of results from different pseudorandom number generators on a common finite sized lattice. For $p \approx p_{\mathrm{c}}$, to first order $R_L(p) \sim 0.5 + k/L + \mathrm{O}\left((p - p_{\mathrm{c}})L^{1/\nu}\right)$ [287], and hence $p_{\mathrm{f}}(L)$ also provides a reasonable estimate of the critical point p_{c}. Ziff and Newman have found that the second order equation $R_L(p_{\mathrm{c}}) \approx 0.5 + kL^{-1} - 0.44L^{-2}$ is a better model of the data at small L [296], however the L^{-2} term is negligible for $L \geq 1024$ at the levels of precision considered here. Values for $p_{\mathrm{f}}(2048)$ were thus obtained from each of the generators described above. These were subsequently compared against each other and against previous

p_c estimates made with the same generators. For large L, $R_L(p)$ rises very steeply in the neighbourhood of $p \approx p_c$. Consequently, $p_f(L)$ is relatively insensitive to the exact value of k provided that $k/L \ll 1$. The uncertainty in k limits the maximum attainable precision in $p_f(2048)$ to $\pm 2 \times 10^{-9}$. This, of course, is essentially the data analysis used in section 3.5, only there with the microcanonical approximation $R_L(p = n/N) \approx R_{L,n}$. Here no such approximation shall be made.

Incidentally, it was observed here that n/N such that $R_{2048,n} = 0.5 + k/L$ (interpolating to non-integer n) usually exceeds $p_f(2048)$ by approximately 2×10^{-8}. This means that the previous estimate of 0.59274603(9) (in subsection 3.5.3, see figure 3.22) should most likely be revised downward to $p_c = 0.5927460(1)$, an adjustment of rather less than the original statistical uncertainty.

The various estimates of $p_f(2048)$ thus obtained are listed in table 3.9. Results are separated according to the generator and initialisation scheme used. SWBb, for instance, indicates the SWB generator with its initialisation list derived from the LCGb output sequence. Similarly, XGx is the XG generator initialised from the WMx output sequence. Results shown are based on surveys of order 10^8 effectively independent samples per occupation level n. Each of these sets involved the generation of order 10^{13} to 10^{14} (approaching 10^{15} in the XG case), pseudorandom numbers.

TTa is the TT generator initialised from LCGa. The TTa based $p_f(2048)$ estimate in table 3.9 is consistent with the results of Newman and Ziff that were also obtained (primarily [285]) from the TT generator (see table 3.3). TTHa is the hashed TT generator, again initialised with LCGa. The TTa and TTHa based estimates are sufficiently different to indicate the probable existence of statistically significant correlations within the TT generator output sequence. This gives cause for concern about the use of the TT generator for this application at this level of precision.

TTTab is the TTT generator with its initial u and v lists constructed by LCGa and LCGb respectively (the two initialising generators being given two different seeds). The TTT based estimate in table 3.9 is consistent with the table 3.3 results of Deng and Blöte most likely obtained from this generator.

QTAa and QTAm are the QTA generator respectively initialised from LCGa and LCGm. Since the QTAa and QTAm results are consistent, there

PRNG	$p_f(2048)$
TTa	0.59274627(11)
TTHa	0.59274588(11)
TTTab	0.59274628(12)
SWBb	0.59274617(17)
QTAa	0.59274588(17)
QTAm	0.59274603(17)
QTBa	0.59274610(17)
QTBb	0.59274621(17)
XGx	0.59274596(15)
MTa	0.59274593(17)
MTm	0.59274585(16)
MTHm	0.59274598(12)
DMTmm	0.59274597(08)

Table 3.9: Site percolation threshold estimates for the square lattice ($p_f(2048)$) obtained by various pseudorandom number generators (PRNGs) as described in the text.

is no evidence that estimates from the QTA generator are especially sensitive to initialisation. The union of these two data sets gives an overall estimate of $p_f(2048) = 0.59274595(12)$ from the QTA generator. This value is consistent with earlier results, in table 3.3, obtained by Ziff and Stell with this generator.

QTBa and QTBb are the QTB generator initialised from LCGa and LCGb respectively. Since the QTBa and QTBb results are consistent, there is no evidence that estimates from the QTB generator are especially sensitive to initialisation. The union of these two data sets gives an overall estimate of $p_f(2048) = 0.59274616(12)$ from the QTB generator. This is consistent with the TTa value, and also with Newman and Ziff's similar observation regarding these two generators [190, 191]. Referring to table 3.3, the value is also consistent with the various threshold estimates obtained by Newman and Ziff using (at least in part) this generator.

MTa is the MT generator initialised from LCGa. MTm is the MT generator initialised from LCGm. Since the MTa and MTm results are consistent, there is no evidence that estimates from the Mersenne twister generator are sensitive to initialisation. The union of these two data sets gives an estimate of $p_f(2048) = 0.59274589(12)$. This is consistent with the MTHm result derived from the hashed generator in table 3.9, and hence there is no evidence

that correlations in the MT output sequence influence the measurement at this level of precision. Hence the MT generator appears to be an adequate choice for the current application. Further combining the MTHm data into the union gives an overall estimate of $p_{\mathrm{f}}(2048) = 0.59274593(8)$ from the MT generator. This MT result is inconsistent with those of the TTT and (unhashed) TT generators. The difference in results with respect to the QTB generator is no more than could be expected by chance in a data set of this size. The SWB, QTA and XG generator based estimates are consistent with that of the MT.

DMTmm is the DMT generator with its two initial lists independently constructed, each from one of a pair of seed words, by LCGm. The DMTmm result is consistent with the combined MT result, thereby indicating that any possible correlations between lower order bits in MT output words are insignificant at this level of precision, or at least no worse than correlations in the higher order bits. This suggests that the single MT generator will be adequate for the purposes of this study. Combining all four Mersenne twister based data sets; MTa, MTm, MTHm, and DMTmm, produces an estimate of $p_{\mathrm{f}}(2048) = 0.59274595(6)$. This is consistent, to within one standard deviation, with the earlier result of section 3.5, obtained there with a combination of MT and DMT in the microcanonical ensemble. The combined value does not alter any of the above conclusions regarding the consistency or otherwise of other generators with the Mersenne twister.

Although details of the procedure used here differs from those of previous works, the results obtained are found to be consistent when the same pseudorandom number generators are used. However, given the use of a consistent method, it has been shown that the results thus obtained can differ with the choice of generator. The level of pseudorandom number generator sensitivity will be method dependent. The spread in results seen here is not extreme as only reasonable quality generators have been used.

The SWB, QTA, QTB, XG and MT generators are backed by strong theory [172, 291, 177, 24] and have been extensively tested elsewhere [291, 266, 149]. Ziff has performed a sensitive hull generating walk test upon several generalised feedback shift register generators [291]. Two-tap generators performed poorly in this test which concluded that they best be avoided for critical applications. Certain quad-tap generators, particularly QTB, performed very well. Analysis indicated that QTB should outperform QTA in

principle, although no obvious problems were observed in the latter. The MT generator has passed tuned collision tests conducted by Tsang, Hui, Chow, Chong and Tso [266]. The LCGa generator failed those same tests. L'Ecuyer and Simard have recently performed thorough randomness tests upon a large assembly of pseudorandom number generators, including SWB, QTB, MT, XG and LCGa [149]. The XG generator passed all tests, the MT failed in a very limited number of instances, QTB and SWB both failed a small number of times, and LCGa failed badly. TT was not specifically tested, although two-tap generators typically performed poorly. Problems with the TT, TTT, and QTA generators have also been noted by Matsumoto and Kurita [176], and the SWB generator is essentially equivalent to a large prime modulus linear congruential generator [261].

Results from the Mersenne twister generator have been consistent under different initialisation methods and effective lattice enumerations (hash functions). With the observation that results from the Mersenne twister differ from those of the two-tap lagged Fibonacci generator, in the presence of evidence suggesting that the two-tap suffers from significant output correlations, and in the absence of evidence for any such correlations in the Mersenne twister output sequence, further Monte Carlo sampling within this chapter shall be performed exclusively with the MT19937 algorithm. Note that results from the SWB, QTA, QTB and XG generators are consistent with those of the MT.

3.6.4 Finite Size Scaling

As before, microcanonical ensemble crossing probability curves, $R_{L,n}$, were determined by Monte Carlo sampling of lattice configurations generated by the algorithms of chapter 2. This sampling involved the generation of well over 10^{15} pseudorandom numbers from the MT19937ar algorithm. The curve for $R_{128,n}$ was determined from some 10^{13} pseudorandom numbers, while the $R_{4096,n}$ curve used 10^{15} pseudorandom numbers. These microcanonical ensemble probabilities were transformed into canonical ensemble crossing probability functions, $R_L(p)$, by the convolution of equation (3.4). Four statistics were then calculated at each L, the median-p critical point estimator $p_{\mathrm{m}}(L)$, the cell-to-cell estimator $p_{\mathrm{cc}}(L)$, the linear combination estimator $p_{\mathrm{h}}(L)$, and the cell-to-site estimator $p_{\mathrm{r}}(L)$.

Ziff's median-p critical point estimator [287], $p_{\rm m}(L)$, is defined such that

$$R_L(p_{\rm m}(L)) = 1/2. \tag{3.20}$$

To second order, the median-p estimator approaches its infinite lattice limit as

$$p_{\rm m}(L) \approx p^*_{\rm m} - aL^{-1-1/\nu} + bL^{-1-\omega-1/\nu}, \tag{3.21}$$

where $\nu = 4/3$ [287, 296], and where Ziff and Newman have determined a value of $\omega = 0.90(2)$ [296] for the scaling exponent originally proposed by Aharony and Hovi [2, 116] (see subsection 3.2.5).

The Reynolds, Stanley and Klein real-space renormalisation group cell-to-cell estimator [211, 212], $p_{\rm cc}(L)$, is defined such that

$$R_L(p_{\rm cc}(L)) = R_{L/2}(p_{\rm cc}(L)). \tag{3.22}$$

The second order scaling relation for the cell-to-cell estimator is given by

$$p_{\rm cc}(L) \approx p^*_{\rm cc} + \frac{a}{\alpha}L^{-1-1/\nu} + cL^{-1-\omega-1/\nu}, \tag{3.23}$$

where $\alpha = 1 - 2^{-1/\nu}$ [296].

The Ziff and Newman linear combination estimator [296], $p_{\rm h}(L)$, is defined as

$$p_{\rm h}(L) \equiv (p_{\rm m}(L) + \alpha p_{\rm cc}(L))/(1+\alpha), \tag{3.24}$$

and was constructed purposely to cancel the lowest order terms of equations (3.21) and (3.23), leaving a much faster approach to the infinite lattice limit. The first order scaling relation is

$$p_{\rm h}(L) \sim p^*_{\rm h} + \frac{b + \alpha c}{1+\alpha}L^{-1-\omega-1/\nu}. \tag{3.25}$$

The real-space renormalisation group cell-to-site fixed point estimator of Reynolds, Klein and Stanley [210], $p_{\rm r}(L)$, is defined such that

$$R_L(p_{\rm r}(L)) = p_{\rm r}(L). \tag{3.26}$$

The second order scaling relation for the fixed point renormalisation group

L	$p_{\rm m}(L)$	$p_{\rm cc}(L)$	$p_{\rm h}(L)$	$p_{\rm r}(L)$
8	0.5842394	0.608314	0.591184	0.6137656
16	0.5898858	0.598828	0.592465	0.6069022
32	0.5918352	0.594825	0.592698	0.6016319
64	0.5924657	0.593413	0.592739	0.5981485
128	0.5926613	0.592952	0.592771	0.5959837
256	0.5927208	0.592808	0.592754	0.5946742

Table 3.10: QTB based Monte Carlo results for site percolation threshold estimators on square lattices of various sizes L. $p_{\rm m}(L)$ is the median-p estimator, $p_{\rm cc}(L)$ is the cell-to-cell estimator, $p_{\rm h}(L)$ is the linear combination estimator, and $p_{\rm r}(L)$ is the cell-to-site estimator. Table is reproduced from Ziff and Newman [296] who obtained these results using their algorithm [190, 191] and the QTB pseudorandom number generator. Uncertainties are generally in the last digit quoted.

estimator is given by

$$p_{\rm r}(L) \sim p_{\rm r}^* + rL^{-1/\nu} + sL^{-2/\nu} \tag{3.27}$$

[296], and so $p_{\rm r}(L)$ has a slower rate of convergence to its infinite lattice limit than the other three estimators above. Each of the four limits $p_{\rm m}^*$, $p_{\rm cc}^*$, $p_{\rm h}^*$ and $p_{\rm r}^*$, should be numerically equivalent to the percolation threshold $p_{\rm c}$, however this more general notation is useful to indicate the origin of any threshold estimates.

Numerical estimates for these quantities are shown in tables 3.10, 3.11 and 3.12. Results of tables 3.11 (obtained here with the Mersenne twister generator only) and 3.12 (including also data from generators found to be consistent with the Mersenne twister in subsection 3.6.3) are consistent with those of table 3.10 (obtained by Ziff and Newman with the QTB generator [296]) where they overlap at $L = 256$ and $L = 128$.

Parametrised fits of equations (3.21), (3.23), (3.25) and (3.27) were made to the data of those three tables. Generally speaking, the quality of the fits deteriorates for $L < 128$, most likely due to large finite size effects. For $L \geq 128$, *i.e.* for the data obtained in this study alone, typically only leading order models were required in order to fit the data. No evidence was found of any $L^{-1-\omega-1/\nu}$ term in the median-p data, for instance. The median-p data can be well fit by an empirical power law, without assuming

the scaling exponent (the fitted exponents are consistent with the assumed value of $-1-1/\nu=-7/4$). The linear combination data is consistent with $p_{\mathrm{cc}}(L)=$ constant, since the limited precision of the data is insufficient to benefit from the improved rate of convergence. Table 3.13 shows the final p_{c} estimates obtained from each of the four generators using either the purely Mersenne twister data of table 3.11 or the combined generators data of table 3.12. Note that individual fits were typically of higher precision than this, with statistical uncertainties as low as $\pm 2\times 10^{-8}$, and that the results shown are robust, making allowances for variations in the data domain and model order of the fit. All results in table 3.13 are in good mutual agreement, with the most precise value being obtained from the median-p estimator.

Consider once again the combined generators median-p data of table 3.12. An empirical power law fit of the form $p_{\mathrm{m}}(L)\sim p_{\mathrm{m}}^{*}-aL^{z}$ finds $p_{\mathrm{m}}^{*}=0.59274598(4)$, $a=0.414(20)$, and $z=-1.75(1)$. The excellent agreement between this model and the experimental data is shown in figure 3.24. The fitted value of z is indistinguishable from the assumed scaling exponent of $-1-1/\nu$. This fit is interesting because it demonstrates the very high data quality achieved in this exercise. The resulting very high precision estimate of p_{c} did not require the assumption of any particular scaling relation and hence ought to be unbiased, accurate and reliable. The limiting factor is going to be the chosen pseudorandom number generators as these have not been tested to the same level of precision as this result.

Considering all the various median-p and cell-to-cell fits conducted in the course of the above, the leading order coefficient a is estimated to have the value 0.415(5). This result is consistent with those of Ziff, Newman, Hovi and Aharony [287, 296, 116] (a here equates to their ratio b_0/a_1). Since k equates to b_0, it follows that $a_1=0.76(1)$ from the data obtained within this exercise. This estimate is also consistent with those of previous works [287, 296, 116].

Taking either the pure Mersenne twister result, $p_{\mathrm{c}}=0.59274596$, or else the combined generators result, $p_{\mathrm{c}}=0.59274598$ (it turns out not to matter which), and returning to the canonical spanning probability curves, a good match between the (respective) $R_L(p)$ data of $128\leq L\leq 4096$ and the theory of $R_L(p_{\mathrm{c}})\approx 0.5+kL^{-1}+\mathrm{O}(L^{-2})$ was had for $k=0.317(1)$. No higher order terms were seen, with the coefficient of L^{-2} being indistinguishable from zero. The value of k found here is a little lower than those of Ziff,

L	$p_{\mathrm{m}}(L)$	$p_{\mathrm{cc}}(L)$	$p_{\mathrm{h}}(L)$	$p_{\mathrm{r}}(L)$
128	0.59266108(21)			0.59598352(23)
256	0.59272062(18)	0.5928085(4)	0.5927460(5)	0.59467466(18)
512	0.59273860(15)	0.5927651(4)	0.5927462(4)	0.59389258(15)
1024	0.59274377(14)	0.5927514(4)	0.5927460(4)	0.59342699(15)
2048	0.59274528(06)	0.5927475(2)	0.5927459(2)	0.59315051(06)
4096	0.59274573(10)	0.5927464(3)	0.5927459(3)	0.59298626(10)

Table 3.11: Mersenne twister based Monte Carlo estimates of several site percolation threshold estimators on square lattices of various sizes. Results were obtained exclusively with the Mersenne twister pseudorandom number generator. Data at $L = 2048$ is the combined MTa, MTm, MTHm, and DMTmm data of table 3.9.

L	$p_{\mathrm{m}}(L)$	$p_{\mathrm{cc}}(L)$	$p_{\mathrm{h}}(L)$	$p_{\mathrm{r}}(L)$
128	0.59266108(21)			0.59598352(23)
256	0.59272062(18)	0.5928085(4)	0.5927460(5)	0.59467466(18)
512	0.59273860(15)	0.5927651(4)	0.5927462(4)	0.59389258(15)
1024	0.59274377(14)	0.5927514(4)	0.5927460(4)	0.59342699(15)
2048	0.59274532(04)	0.5927476(1)	0.5927459(1)	0.59315055(04)
4096	0.59274573(10)	0.5927463(2)	0.5927459(3)	0.59298626(10)

Table 3.12: Mixed generator based Monte Carlo estimates for several site percolation threshold estimators on square lattices of various sizes. Results are those of table 3.11 (purely Mersenne twister based) merged with the SWB, QTA, QTB and XG data of table 3.9 (at $L = 2048$).

Limit	Mersenne	Combined
p_{r}^*	0.592745(1)	0.592745(1)
p_{cc}^*	0.5927459(2)	0.5927459(2)
p_{h}^*	0.59274596(7)	0.59274598(6)
p_{m}^*	0.59274596(4)	0.59274598(3)

Table 3.13: Extrapolated infinite lattice limit estimates for the percolation threshold. Results are shown, by estimator, for the Mersenne twister only data of table 3.11 (MT, MTH, DMT), and also for the combined generators data of table 3.12 (MT, MTH, DMT, SWB, QTA, QTB, XG).

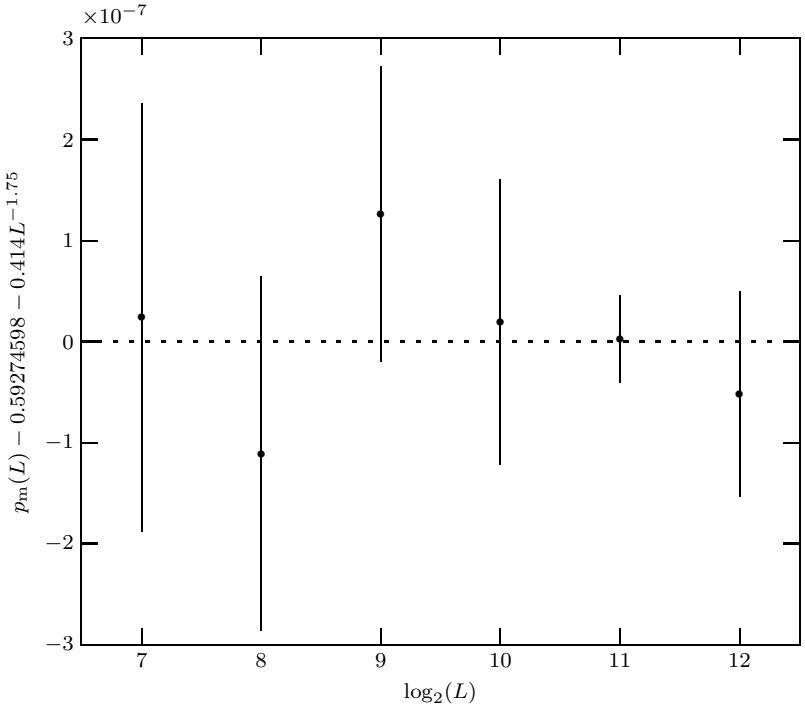

Figure 3.24: Empirical power law model, $p_{\mathrm{m}}(L) = p^*_{\mathrm{m}} - aL^z$, fitted to the combined generators median-p data, $p_{\mathrm{m}}(L)$, of table 3.12. Error bars are one standard deviation statistical sampling uncertainties.

$k = 0.319(1)$ [287], and Newman and Ziff, $k = 0.320(1)$ [191]. The difference in $p_{\mathrm{f}}(2048)$ resulting from using $k = 0.317$, as opposed to $k = 0.320$, in equation (3.19) is around 6×10^{-9}. This is much less than the statistical uncertainties in the results of table 3.9, upholding the claimed insensitivity of those estimates to k. Hence those results remain reasonable (generator biased) estimates for p_{c}, and the direct comparison with earlier p_{c} estimates is valid.

3.6.5 Final Threshold Estimate

Several estimates have now been made for the square site percolation threshold, p_{c}, with one standard deviation statistical uncertainties as low as 2×10^{-8}. Although it is common practice to quote the 'best', *i.e.* the most precise, result obtained from a set of measurements, this should not be done at the expense of accuracy. Given the very large quantities of pseudorandom numbers used here, the precision limiting factor could well be the quality of the chosen generators rather than statistical sampling fluctuations. There is evidence to suggest that earlier results have indeed been influenced by the generators used. While it would probably be worthwhile to repeat these measurements with the very high quality xor4096 [24] or WELLS [197] generators, the Mersenne twister appears adequate for the present task. However, all deterministic generators must eventually fail tests of randmoness, and hence there will be an ongoing requirement for improved algorithms as computers become faster and rates of pseudorandom number generation increase [40]. Although the generators used here have all been tested, the application specific testing was not performed to the same level of precision as the later p_{c} estimates. The Mersenne twister generator was tested more thoroughly than the others, and although these were found to be consistent with the twister, there remains a somewhat subjective decision on whether or not to use them. There has also been some argument in the literature, on and off, regarding the forms of the finite size scaling expressions (particularly equation (3.21) [287, 2, 116]), although these are now largely resolved [240, 296] and, in any case, the data quality obtained here both supports the assumed relations and renders their assumption unnecessary. Bearing all of this in mind, this study finds that a precise and defensible estimate

for the square site percolation threshold is:

$$p_c = 0.59274598(4) \tag{3.28}$$

This conservative result offers a significant increase in precision over any earlier measurement and, although the precautions of subsection 3.6.3 cannot absolutely rule out possible generator bias, they do suggest that this estimate should be accurate. The result is consistent with the preliminary estimate of section 3.5 and also (on a case-by-case basis) with the majority of individual estimates in table 3.3. Taken collectively however, the results of table 3.3 indicate a threshold value of around $p_c = 0.5927463(1)$, and the result of equation (3.28) is inconsistent with that figure. It appears, from the analysis conducted here, that the difference is likely attributable to the various pseudorandom number generators used. Consequently, the result of equation (3.28) is ostensibly reliable, and the paper [153] already has citations [102]. Indeed, this estimate has since been independently confirmed by Feng, Deng, and Blöte [66], using both a transfer matrix technique ($p_c = 0.59274605(3)$) and a Monte Carlo calculation ($p_c = 0.59274606(9)$) based on the TTT generator. While the results of subsection 3.6.3 suggest that use of the TTT generator may lead to an overestimate of p_c, this remains consistent with a comparison between the results of [66] and equation (3.28). In any case, the Feng, Deng, and Blöte results were not calculated with the same closed lattice boundary conditions under which generators were tested here.

3.7 Summary

This chapter has demonstrated the utility of the algorithms of the previous chapter by successfully applying them to the two-dimensional square-lattice site-percolation model. The algorithms were employed in an application specific test of pseudorandom number generator quality, and the results of this test allowed both the identification of a suitable generator for high precision work, and also the detection of poorer generators that may have biased earlier works. By obtaining good statistics from large lattices, finite size effects were reduced to the point that higher order terms of scaling relations were negligible and the exponents of leading order terms could be verified em-

pirically. The net result of all this has been the first accurate measurement of the square site percolation threshold to achieve seven significant digits of precision. In comparison, the first six significant digits have been known for twenty years. The extra digit is useful since many studies of critical phenomena of the percolation model require knowledge of the critical point and are highly sensitive to its assumed value. In addition, so long as this *bona fide* mathematical problem remains unsolved, Monte Carlo results, such as this, provide the only means for testing conjectures.

PHYSICAL REVIEW E **78**, 031131 (2008)

Pseudo-random-number generators and the square site percolation threshold

Michael J. Lee
Department of Physics and Astronomy, University of Canterbury, Christchurch, New Zealand
(Received 3 July 2008; published 25 September 2008)

Selected pseudo-random-number generators are applied to a Monte Carlo study of the two-dimensional square-lattice site percolation model. A generator suitable for high precision calculations is identified from an application specific test of randomness. After extended computation and analysis, an ostensibly reliable value of $p_c=0.592\,745\,98(4)$ is obtained for the percolation threshold.

DOI: 10.1103/PhysRevE.78.031131 PACS number(s): 64.60.ah, 02.70.Uu

I. INTRODUCTION

The square lattice site percolation threshold, p_c, is a clearly and simply defined mathematical concept [1,2]. Percolation models have been well studied, and are known for their numerous applications [3]. Yet to date, no analytical expression has been found for the numerical value of p_c. The square site lattice lacks the symmetry that has allowed exact solutions on other topologies [2,4–9]. So long as the problem remains intractable, statistical estimates from Monte Carlo studies can, at least, offer approximate values. Such calculations invariably make extensive use of some form of pseudorandom-number generator (PRNG).

A PRNG is a deterministic algorithm that outputs a sequence of words with properties closely mimicking those of a truly random sequence. Well analyzed generators include the linear congruential, lagged Fibonacci, generalized feedback shift register, and derivatives thereof [10–24]. Because these algorithms are simple, they do not produce output with the complexity of a random sequence [25,26]. The autocorrelation coefficients of a pseudorandom sequence are not identically zero, and these departures from true randomness introduce a sampling bias that leads to systematic error.

Twenty years ago, concern was being given to the demands then made of PRNGs in calculations using 10^{12} pseudorandom numbers generated at MHz rates [27]. Recently, high performance parallel computer systems with thousands, rather than tens or hundreds, of processors have become much more widely available. These enable calculations with 10^{15} pseudorandom numbers generated at GHz rates, and are likely to play a central role in future research. Very high precision can now be achieved through brute force of sampling, but accuracy is another matter. For reliable Monte Carlo estimates at these new higher precision levels, the PRNG(s) chosen must be of sufficient quality. Hence contemporary demands upon PRNGs are, and will continue to become, much greater than in the past.

This study compares several established PRNGs within the context of the square site percolation problem. Following application specific testing, a seemingly reliable generator is identified. This is subsequently used to locate the percolation threshold with, in principle, both accuracy and precision.

II. GENERATORS

Throughout this study, the computational word length, w, shall be fixed at 32. All arithmetical operations taking place within any PRNG are performed in modulo 2^w. All PRNG arithmetical operands, and products thereof, are members of $\{0:2^w-1\}$, where $\{a:b\}$ denotes the set of all integers not less than a and not greater than b. Consequently the words of any PRNG output sequence also belong to $\{0:2^w-1\}$. The ith word of an output sequence shall be denoted by x_i. With one noted exception, no output sequence is decimated in any way. The first million words of each sequence are discarded prior to beginning any Monte Carlo sampling procedure.

Some PRNGs make use of bitwise operations within their internal algorithms. The notation adopted here is $\oplus$ for bitwise Boolean logical exclusive or, and $\triangleright m$ for shift m bits to the right (where m is a positive integer). Whenever these bitwise operations are performed, the operands are decomposed into their respective standard binary representations, most-significant bit (leftmost) to least-significant bit (rightmost). Arithmetic being constrained to a subset of the integers, any bits shifted to a position right of the decimal point are lost. Hence, within this study, the operation of $\triangleright m$ is equivalent to integer division by 2^m.

The specific PRNGs considered within this exercise are defined as follows.

TT is the two-tap additive lagged Fibonacci generator $x_i = x_{i-418}+x_{i-1279}$. This generator has previously been used for high-precision percolation threshold measurement by Newman and Ziff [28,29].

TTT combines the output from a pair of two-tap generalized feedback shift-register generators, $u_i=u_{i-471}\oplus u_{i-9689}$ and $v_i=v_{i-30}\oplus v_{i-127}$, to return a single word $x_i=u_i\oplus v_i$. This is the generator most likely used for two- and three-dimensional percolation by Deng and Blöte [30,31].

SWB is a Marsaglia and Zaman subtract with borrow generator, $x_i=x_{i-222}-x_{i-237}-\beta_{i-1}$, where the borrow, β_i, is equal to 1 if $x_{i-222}<x_{i-237}+\beta_{i-1}$, and is otherwise equal to zero [16,20].

QTA is the quad-tap generalized feedback shift-register generator $x_i=x_{i-157}\oplus x_{i-314}\oplus x_{i-471}\oplus x_{i-9689}$, as used by Ziff and Stell [32,33] (see [17]).

QTB is the quad-tap generalized feedback shift-register generator $x_i=x_{i-471}\oplus x_{i-1586}\oplus x_{i-6988}\oplus x_{i-9689}$. This generator has been used by Newman and Ziff, and has been found to produce threshold estimates consistent with those of the TT generator [28,29,34].

XG is Brent's xor4096 generalized Marsaglia xorshift generator [21,23]. Specifically, the implementation xor4096i, from Brent's C language xorgens304 distribution, was that

used here. This generator has performed well in randomness tests conducted by L'Ecuyer and Simard [24].

MT is Matsumoto and Nishimura's MT19937 Mersenne twister generator [18]. Specifically, their MT19937ar C language distribution was the implementation used here. The MT19937 algorithm has been used for computing integrals in semirigorous work by Balister, Bollobás, Walters, and Riordan [35,36], and for Monte Carlo sampling by Lee [37].

DMT is a pair of MT generators operated entirely independently of one another. The output sequence from each of these generators is decimated, with only every fourth word used. Lattice sites are then selected by means of their Cartesian coordinates, using one number from each generator. This scheme has previously been used by Lee [37].

Let L be the number of sites lying on each edge of an $L\times L$ square lattice. Every site on that lattice is typically given a unique index or label, $j\in\{0:L^2-1\}$. In a microcanonical ensemble Monte Carlo calculation [28,29,34], such as those performed here, PRNG output words, x_i, are used to pseudorandomly select sites, s_j, for occupation. Now, consider a transformation $T(x,N)\equiv x\triangleright(w-\log_2 N)$, where N is a positive-integer power of 2. This is a distribution preserving many-to-one surjective map from the integers $x\in\{0:2^w-1\}$ to the integers $T(x,N)\in\{0:N-1\}$. Further consider a bijection H that maps the integers $\{0:N-1\}$ onto site labels. With the usual choice of site labels also being the integers $\{0:N-1\}$, H is conventionally taken to be the identity map. In the single generator systems defined above, PRNG output word x_i is associated with site $s_{j(x_i)}$ via $j(x_i)=H[T(x_i,L^2)]$. For the DMT, a pair of PRNG output words, u_i and v_i (one from each of the output decimated MT generators), is mapped to site $s_{j(u_i,v_i)}$ via $j(u_i,v_i)=H[T(u_i,L)+LT(v_i,L)]$. This halves the number of bits actually used from each output word (the most significant bits being those retained).

Each of these generators must be provided with an initial finite sequence of words from which to begin calculating an infinite pseudorandom sequence. In the case of the TT generator for instance, a list of some 1279 initial words is required. These initial lists were constructed by one of four simpler generators, here denoted LCGa, LCGb, LCGm, and WMx. LCGa is the linear congruential generator, $x_i=69\,069x_{i-1}+1$, suggested by Marsaglia [12]. LCGb is a similar linear congruential generator, $x_i=69\,069x_{i-1}+1\,234\,567$, also due to Marsaglia [20]. LCGm is the modified linear congruential generator, $x_i=1\,812\,433\,253[x_{i-1}\oplus(x_{i-1}\triangleright 30)]+i$, appearing in Matsumoto and Nishimura's MT19937ar distribution of their Mersenne twister algorithm. WMx is the Weyl modified Marsaglia xorshift generator built into the xor4096i algorithm appearing within Brent's xorgens304 distribution [21,23]. These initialization generators are themselves seeded from a single word, $x_0\in\{0:2^w-1\}$. The TTT and DMT generators both require two initialization lists, each being derived from one of these four generators, each starting with a distinct independent seed word. Single word initialization does limit the number of different sequences that can be obtained from a given combination of main generator and initialization generator to 2^w. This is not a problem here since the total number of sequences generated was five orders of magnitude less than this, and those were divided between many different generator combinations.

III. TEST PROCEDURE

The above listed generators were compared, in the context of site percolation on the square lattice, by using each one to make a Monte Carlo estimate of the crossing probability function, $R_{L,n}$, at $L=2048$, over the domain $n\in\{2\,474\,000:2\,498\,300\}$. $R_{L,n}$ is defined as the probability that a single cluster connects two specified opposing boundary sides of the $N=L\times L$ square lattice in the microcanonical ensemble when precisely n random sites are occupied. The value of $R_{2048,n}$ monotonically increases from around 0.05 at $n=2\,474\,000$ to around 0.95 at $n=2\,498\,300$. Hence, the occupation domain studied encompasses the critical region of the percolative phase transition. The numerous lattice configurations required to accurately determine $R_{L,n}$ were constructed, from each PRNG output sequence, over the above domain only, by the unbiased algorithm of Lee [37]. The only exception was the XG generator from which samples were obtained over the same domain by the unbiased algorithm of Newman and Ziff [28,29].

The Newman and Ziff binomial convolution

$$R_L(p)=\sum_n \binom{N}{n} p^n(1-p)^{N-n}R_{L,n} \tag{1}$$

then gives the crossing probability, $R_L(p)$, in the canonical ensemble where each lattice site is independently randomly occupied at probability p [28,29,34]. In principle the summation should run over all $n\in\{0:N\}$, but as samples were taken only over the restricted domain of n above, the summation was truncated accordingly. The standard deviation of the binomial distribution in Eq. (1) is given by $\sigma_{L,p}\approx L\sqrt{p(1-p)}$. The data analysis here is concerned with values of p such that the distribution maximum, located at $n=\mathrm{nint}(pN)$, lies between 10 and $12\sigma_{L,p}$ from the nearest end of the sampling region. Consequently, the truncation induced error in $R_L(p)$ is not more than 10^{-15}. This is completely negligible when compared to statistical sampling uncertainties, which were never less than 10^{-8}.

The canonical crossing probability curve is used to identify a site occupation probability, $p_f(L)$, defined such that

$$R_L[p_f(L)]=1/2+k/L, \tag{2}$$

where $k=0.320(1)$, as determined by Ziff and Newman [34] (their parameter b_0). For $p\approx p_c$, to first order $R_L(p)\sim 0.5+k/L+O[(p-p_c)L^{1/\nu}]$ [33], and hence $p_f(L)$ provides a reasonable estimate of the critical point p_c. Ziff and Newman have found that the second-order equation $R_L(p_c)\approx 0.5+kL^{-1}-0.44L^{-2}$ is a better model of the data at small L [34], however the L^{-2} term is negligible for $L\geqslant 1024$ at the levels of precision considered here. Values for $p_f(2048)$ were thus obtained from each of the PRNGs described above. These were subsequently compared against each other and against previous p_c estimates made with the same generators. For large L, $R_L(p)$ rises very steeply in the neighborhood of $p\approx p_c$. Consequently, $p_f(L)$ is relatively insensitive to the exact value of k provided that $k/L\ll 1$. The uncertainty in k limits the maximum attainable precision in $p_f(2048)$ to $\pm 2\times 10^{-9}$.

$$\begin{array}{cccc} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{array} \qquad \begin{array}{cccc} s_0 & s_{11} & s_6 & s_1 \\ s_{12} & s_7 & s_2 & s_{13} \\ s_8 & s_3 & s_{14} & s_9 \\ s_4 & s_{15} & s_{10} & s_5 \end{array}$$

FIG. 1. Standard (left-hand side) and (example) nonstandard (right-hand side) enumerations of the $L=4$ square lattice.

Combinatorial terms of the binomial distribution in Eq. (1) were calculated by the essentially exact method of Newman and Ziff [29]. For p near p_c, use of the Gaussian approximation to the binomial would have introduced an error of order 10^{-8} in $R_{2048}(p)$, this corresponding to an error of order 10^{-10} in p itself. It is sometimes possible to dispense with the convolution altogether and make a microcanonical ensemble approximation of $R_L(p=n/N)\approx R_{L,n}$. With $L=2048$, and for p near p_c, this introduces an error of around 4×10^{-6} in $R_L(p)$, which corresponds to an error of around 2×10^{-8} in p. This approximation is acceptable at low enough precision, has the advantage that only a much narrower domain of sampling need be considered, and has been employed in earlier work by Lee [37]. However, since the induced error [measured as the difference between p and n/N such that either $R_L(p)=R_{L,n}=0.5$ or $R_L(p)=R_{L,n}=0.5+k/L$] was found to scale as only $L^{-1.5(3)}$, when a set of measurements are to be taken over a range of lattice sizes, to precisions of order $1/N$, the error will become significant at large L. The approximation was not adopted here.

Correlations inevitably found in the output sequence of any deterministic pseudo-random-number generator will result in correlations within the spatial pattern of occupied sites upon the lattice. As noted by Compagner [38], this in turn will bias the resulting Monte Carlo estimate of the crossing probability function. Consequently, when estimates obtained from two different generators are inconsistent, then at least one of those generators likely suffers from significant correlations in its output sequence, hence rendering it unsuitable for use at the level of precision of the study. Because the true values of $R_{L,n}$, $R_L(p)$, and p_c are unknown, it will be unclear as to which of the generators is deficient.

While there is merit in performing general tests on PRNGs, it is often preferable to have an application specific test such as the sensitive hull walk of Ziff [17]. Here a scheme is used that changes the relation between numerical PRNG output sequences and the spatial patterns of occupied lattice sites, without altering the underlying problem or topology in any way. The standard enumeration of the lattice, as shown in Fig. 1, prescribes a specific relation between patterns in PRNG output and in clusters of occupied sites. By adopting some other (nonstandard) enumeration, as per the example in Fig. 1, some different relation is obtained. An ideal random-number generator will produce results independent of the chosen enumeration. A pseudo-random-number generator, with correlations in its output sequence, will produce results that do depend upon the enumeration. By comparing results from a common generator on two different lattice enumerations, inadequate, outcome biasing, generators may be identified. This simple application specific test does not require knowledge of the percolation threshold or spanning probability curves.

The direct approach to implementing such an enumeration is to allocate each site a set of pointers explicitly identifying its geometrical neighbors. In the nonstandard enumeration of Fig. 1, for example, s_7 would have pointers to s_2, s_3, s_{11}, and s_{12}. Sites are pseudorandomly selected as per normal and the Monte Carlo sampling proceeds just as for the standard enumeration. In practice this results in a dramatic performance decrease of the simulations (more than a factor of 2 was found in this study). The problem is believed to be the cache prefetching of the high performance computer system used, where if, in some linear array, s_j is being accessed then the hardware assumes that s_{j+1} (being the next contiguous data element in memory) will be wanted next.

An alternative method is to change the mapping, H, between scaled PRNG output words, $y=T(x,N)$, and site labels, j. In the standard enumeration of Fig. 1, $H(y)=y$ is the identity map. In the nonstandard enumeration, $H(0)=0$, $H(11)=1$, $H(6)=2$, and so on, with the general relation being $H(y)=3y \pmod{16}$. This is analogous to the cluster-label labels of Hoshen and Kopelman [39]. The nonstandard enumeration is that effectively in use, while the standard enumeration is preserved in computer memory, thus avoiding performance problems.

When the hash function, H, is simple (that is, of similar algebraic complexity to the PRNG), it will not so much hide correlations in the output sequence as manifest those patterns in some other way, giving rise to a different estimate for $p_f(L)$. If, on the (systematic) standard enumeration, correlations in PRNG output lead to spatial correlations of occupied sites that in turn bias the estimate, then, on some other (systematic) nonstandard enumeration, those same PRNG correlations will give rise to spatial correlations of a different nature that bias the estimate in some other way. This provides a simple, application specific, test for PRNG biasing of the Monte Carlo samples. If a given PRNG is correlation free, then the estimates derived from it will be independent of the lattice enumeration. If instead, the PRNG output does suffer from correlations, then different enumerations may lead to different results. The test can be tuned, with the hash chosen so as to maximize the observed shift in the test quantity. A simple hash related to the taps or period is bound to highlight intrinsic PRNG shortcomings [38]. Alternatively, a hash much more complex than the generator algorithm could go some way toward hiding output sequence correlation induced bias.

Hence define two further generators, TTH and MTH, as (respectively) the exact same TT and MT generators defined previously, but with a somewhat arbitrary nontrivial mapping $H(y)=947y \pmod{N}$ between integers $y\in\{0:N-1\}$ and site labels $j\in\{0:N-1\}$. On a lattice of $L=2048$, this hash is equivalent to a systematic nonstandard enumeration where the rightward and downward neighbors of site s_j are (when they exist) $s_{j+3\ 171\ 195 (\mathrm{mod}\ N)}$ and $s_{j+1\ 824\ 768 (\mathrm{mod}\ N)}$, respectively.

IV. TEST RESULTS

The various estimates of $p_f(2048)$ thus obtained are listed in Table I. Results are separated according to the generator and initialization scheme used. SWBb, for instance, indicates the SWB generator with its initialization list derived from the

TABLE I. Site percolation threshold estimates for the square lattice [$p_f(2048)$] obtained by various pseudo-random-number generators (PRNGs) as described in the text.

PRNG	$p_f(2048)$
TTa	0.592 746 27(11)
TTHa	0.592 745 88(11)
TTTab	0.592 746 28(12)
SWBb	0.592 746 17(17)
QTAa	0.592 745 88(17)
QTAm	0.592 746 03(17)
QTBa	0.592 746 10(17)
QTBb	0.592 746 21(17)
XGx	0.592 745 96(15)
MTa	0.592 745 93(17)
MTm	0.592 745 85(16)
MTHm	0.592 745 98(12)
DMTmm	0.592 745 97(08)

LCGb output sequence. Similarly, XGx is the XG generator initialized from the WMx output sequence. Results shown are based on surveys of order 10^8 effectively independent samples per occupation level n. Each of these sets involved the generation of order 10^{13} to 10^{14} (approaching 10^{15} in the XG case), pseudorandom numbers.

TTa is the TT generator initialized from LCGa. The TTa based $p_f(2048)$ estimate in Table I is consistent with the results of Newman and Ziff that were also obtained (primarily [40]) from the TT generator (see Table II). TTHa is the hashed TT generator, again initialized with LCGa. The TTa and TTHa based estimates are sufficiently different to indicate the probable existence of statistically significant correlations within the TT generator output sequence. This gives cause for concern about the use of the TT PRNG for this application at this level of precision.

TTTab is the TTT generator with its initial u and v lists constructed by LCGa and LCGb, respectively (the two initializing generators being given two different seeds). The TTT-based estimate in Table I is consistent with the Table II results of Deng and Blöte most likely obtained from this generator.

QTAa and QTAm are the QTA generators respectively, initialized from LCGa and LCGm. Since the QTAa and QTAm results are consistent, there is no evidence that estimates from the QTA generator are especially sensitive to initialization. The union of these two data sets gives an overall estimate of $p_f(2048)=0.592\,745\,95(12)$ from the QTA generator. This value is consistent with earlier results, in Table II, obtained by Ziff and Stell with this generator.

QTBa and QTBb are the QTB generator initialized from LCGa and LCGb, respectively. Since the QTBa and QTBb results are consistent, there is no evidence that estimates from the QTB generator are especially sensitive to initialization. The union of these two data sets gives an overall estimate of $p_f(2048)=0.592\,746\,16(12)$ from the QTB generator. This is consistent with the TTa value, and also with the Newman and Ziff similar observation regarding these two generators [28,29]. Referring to Table II, the value is also consistent with the various threshold estimates obtained by Newman and Ziff using (at least in part) this generator.

MTa is the MT generator initialized from LCGa. MTm is the MT generator initialized from LCGm. Since the MTa and MTm results are consistent, there is no evidence that estimates from the Mersenne twister generator are sensitive to initialization. The union of these two data sets gives an estimate of $p_f(2048)=0.592\,745\,89(12)$. This is consistent with the MTHm result derived from the hashed generator in Table I, and hence there is no evidence that correlations in the MT output sequence influence the measurement at this level of precision. Hence, the MT generator appears to be an adequate choice for the current application. Further combining the MTHm data into the union gives an overall estimate of $p_f(2048)=0.592\,745\,93(8)$ from the MT generator. This MT result is inconsistent with those of the TTT and (unhashed) TT generators. The difference in results with respect to the QTB generator is no more than could be expected by chance in a data set of this size. The SWB, QTA and XG generator-based estimates are consistent with that of the MT.

DMTmm is the DMT generator with its two initial lists independently constructed, each from one of a pair of seed words, by LCGm. The DMTmm result is consistent with the combined MT result, thereby indicating that any possible correlations between lower order bits in MT output words are insignificant at this level of precision, or at least no worse than correlations in the higher-order bits. This suggests that the single MT generator will be adequate for the purposes of this study. Combining all four Mersenne twister-based data sets; MTa, MTm, MTHm, and DMTmm, produces an estimate of $p_f(2048)=0.592\,745\,95(6)$. This is consistent with the result of Lee, in Table II, obtained with this same mixture of generators but in the microcanonical approximation $R_L(n/N)\approx R_{L,n}$. The combined value does not alter any of the above conclusions regarding the consistency or otherwise of other generators with the Mersenne twister. Regarding the previous estimate of Lee, it was observed here that n/N, such that $R_{2048,n}=0.5+k/L$ (interpolating to noninteger n), usually exceeds $p_f(2048)$ by approximately 2×10^{-8}. That being so, a revised estimate of the presented result would be $p_c=0.592\,7460(1)$. This adjustment is much smaller than statistical uncertainties.

Although the procedure used here differs from those of previous works, the results obtained are found to be consistent when the same pseudo-random-number generators are used. However, given the use of a consistent method, it has been shown that the results thus obtained can differ with the choice of generator. The level of PRNG sensitivity will be method dependent. The spread in results seen here is not extreme as only reasonable quality generators have been used.

A great deal of theoretical work has gone into developing these classes of generator, and extensive general tests have been made of them elsewhere. Ziff has performed a sensitive hull generating walk test upon several generalized feedback shift register generators [17]. Two-tap generators performed poorly in this test which concluded that they best be avoided for critical applications. Certain quad-tap generators, particu-

TABLE II. Estimates of the square site percolation threshold presented in the literature. The pseudo-random-number generator(s) used are given where known. Generator T is a Tausworthe generator, while C is a congruential generator. TTT is the generator most likely used by Deng and Blöte. References are provided for both the result and the generator whenever those come from different sources. Uncertainties are quoted as one standard deviation statistical errors, except in the semirigorous results of Balister, Bollobás, and Walters (99.99% confidence bound) and of Riordan and Walters (99.9999% confidence bound). Only those results derived from currently accepted scaling relations are shown from the greater collection in Hu, Chen, and Wu. This table is essentially a continuation of that appearing in Ziff and Sapoval [41], there going back to 1960.

Year	Reference	Author(s)	Method	Generator(s)	Result
1986	[41]	Ziff and Sapoval	Hull gradient	T	0.592 745(2)
1988	[17,32]	Ziff and Stell	Hull gradient	QTA	0.592 746 0(5)
1989	[42]	Yonezawa, Sakamoto, and Hori	Planar crossing		0.593 0(1)
1992	[33]	Ziff	Hull crossing	QTA	0.592 746 0(5)
1994	[43]	Hu	Histogram Monte Carlo		0.592(8)
1995	[44]	Hu	Histogram Monte Carlo		0.592 8(1)
1996	[45]	Hu, Chen, and Wu	Histogram Monte Carlo		0.592 78(2)
1996	[45]	Hu, Chen, and Wu	Histogram Monte Carlo		0.592 83(4)
1996	[45]	Hu, Chen, and Wu	Histogram Monte Carlo		0.592 67(6)
1996	[45]	Hu, Chen, and Wu	Histogram Monte Carlo		0.581 4(30)
1996	[45]	Hu, Chen, and Wu	Histogram Monte Carlo		0.604 1(30)
2000	[28,29]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.592 746 21(13)
2000	[28,29]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.592 746 36(14)
2000	[28,29]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.592 746 06(15)
2000	[28,29]	Newman and Ziff	Toroidal wrapping	TT, QTB	0.592 746 29(20)
2000	[28,40]	Ziff	Hull gradient	QTB	0.592 746 5(2)
2002	[34]	Ziff and Newman	Planar crossing	QTB	0.592 746 4(5)
2003	[46,47]	Martins and Plascak	Toroidal wrapping	C	0.592 7(1)
2003	[46,47]	Martins and Plascak	Toroidal wrapping	C	0.592 9(3)
2005	[30,31]	Deng and Blöte	Cylindrical correlation	TTT	0.592 746 5(4)
2005	[30,31]	Deng and Blöte	Cylindrical correlation	TTT	0.592 746 6(6)
2005	[30,31]	Deng and Blöte	Cylindrical correlation	TTT	0.592 746 6(8)
2005	[30,31]	Deng and Blöte	Cylindrical correlation	TTT	0.592 746 8(10)
2005	[35]	Balister, Bollobás and Walters	Semirigorous	MT	0.592 7(8)
2007	[36]	Riordan and Walters	Semirigorous	MT	0.592 75(25)
2007	[37]	Lee	Planar crossing	MT, DMT	0.592 746 03(9)

larly QTB, performed very well. Analysis indicated that QTB should outperform QTA in principle, although no obvious problems were observed in the latter. Problems with generators in the same class as TT, TTT, and QTA have also been noted by Matsumoto and Kurita [48]. The MT generator has passed tuned collision tests conducted by Tsang, Hui, Chow, Chong, and Tso [22]. The LCGa generator failed those same tests. L'Ecuyer and Simard have recently performed thorough randomness tests upon a large assembly of PRNGs, including SWB, QTB, MT, XG, and LCGa [24]. The XG generator passed all tests, the MT failed in a very limited number of instances, QTB and SWB both failed a small number of times, and LCGa failed badly. TT was not specifically tested, although two-tap generators typically performed poorly. Tezuka, L'Ecuyer, and Couture have shown that generators of the SWB class are essentially equivalent to large prime modulus linear congruential generators and so can be unreliable [49]. New generators named WELL, with improved output sequence properties over the MT generator, have recently been devised by Panneton, L'Ecuyer, and Matsumoto [50].

As noted by Ferrenberg, Landau, and Wong, it is highly desirable to have algorithm and application specific tests of pseudo-random-number generators, regardless of any general tests that the generator may have passed [51]. Within this exercise, results from the Mersenne twister generator have been consistent under different initialization methods and effective lattice enumerations (hash functions). With the observation that results from the Mersenne twister differ from those of the two-tap lagged Fibonacci generator, in the presence of evidence suggesting that the two-tap suffers from significant output correlations, and in the absence of evidence for any significant correlations in the Mersenne twister output sequence, further Monte Carlo sampling within this exercise shall be performed exclusively with the MT19937 algorithm. Note that results from the SWB, QTA, QTB, and

TABLE III. Site percolation threshold estimators on square lattices of various sizes L. $p_m(L)$ is the median-p estimator, $p_{cc}(L)$ is the cell-to-cell estimator, $p_h(L)$ is the linear combination estimator, and $p_r(L)$ is the fixed point estimator. Results were obtained with the Mersenne twister pseudorandom number generator.

L	$p_m(L)$	$p_{cc}(L)$	$p_h(L)$	$p_r(L)$
128	0.592 661 08(21)			0.595 983 52(23)
256	0.592 720 62(18)	0.592 808 5(4)	0.592 746 0(5)	0.594 674 66(18)
512	0.592 738 60(15)	0.592 765 1(4)	0.592 746 2(4)	0.593 892 58(15)
1024	0.592 743 77(14)	0.592 751 4(4)	0.592 746 0(4)	0.593 426 99(15)
2048	0.592 745 28(06)	0.592 747 5(2)	0.592 745 9(2)	0.593 150 51(06)
4096	0.592 745 73(10)	0.592 746 4(3)	0.592 745 9(3)	0.592 986 26(10)

XG generators are consistent with those of the MT. Also, it might be interesting to conduct future work with the high quality XG [23] or WELL [50] generators.

V. THRESHOLD DETERMINATION

Having identified the Mersenne twister as a suitable PRNG for the problem, a more precise determination of the square site percolation threshold can now be made. This will be based upon Monte Carlo estimates of the microcanonical $R_{L,n}$ curves for $128 \leqslant L \leqslant 4096$ (a span of some three orders of magnitude in N).

Data for $L \leqslant 1024$ was obtained exclusively from the MT generator initialized by LCGm, and Monte Carlo sampling was conducted with the algorithm of Lee [37]. Sampling domains were $n \in \{8900:10\,500\}$ on the $L=128$ lattice, $n \in \{37\,300:40\,400\}$ on the $L=256$ lattice, $n \in \{152\,300:158\,500\}$ on the $L=512$ lattice, and $n \in \{615\,500:627\,600\}$ on the $L=1024$ lattice. The data at $L=2048$ is the combined MTa, MTm, MTHm, and DMTmm data from Table I. As noted, that data was obtained with the same algorithm over the site occupation domain $n \in \{2\,474\,000:2\,498\,300\}$. Due to hardware constraints, the $L=4096$ data was obtained with the more memory efficient algorithm of Newman and Ziff [28,29]. For this algorithm, the entire domain, $n \in \{0:N\}$, is sampled, however observations were made only for $n \in \{9\,920\,000:9\,969\,000\}$. Once again, the LCGm initialized MT generator was used. Lattices of L much more than 4096 could not be accommodated by the computer system used without substantial decreases in performance. Estimates at each L are based upon between 1×10^8 (at $L=4096$) and 4×10^9 (at $L=128$) independent samples per occupation level, n. These required the generation of between 10^{13} (at $L=128$) and 10^{15} (at $L=4096$) pseudorandom numbers.

As before, these microcanonical ensemble crossing probability curves, $R_{L,n}$, were transformed into canonical ensemble crossing probability functions, $R_L(p)$, by the convolution of Eq. (1). Because the various microcanonical sampling domains all encompass $\pm 12\sigma_{L,p}$ of the convolution region about the critical point, the domain restriction induced error in $R_L(p)$ is completely negligible for the values of p considered here.

Several statistics were calculated from each $R_L(p)$ curve. These were Ziff's median-p critical point estimator [33], $p_m(L)$, defined such that

$$R_L[p_m(L)] = 1/2, \tag{3}$$

the Reynolds, Stanley, and Klein real-space renormalization group cell-to-cell estimator [52,53], $p_{cc}(L)$, defined such that

$$R_L[p_{cc}(L)] = R_{L/2}[p_{cc}(L)], \tag{4}$$

the Ziff and Newman linear combination estimator [34], $p_h(L)$, defined as

$$p_h(L) \equiv [p_m(L) + \alpha p_{cc}(L)]/(1+\alpha), \tag{5}$$

where $\alpha \equiv 1 - 2^{-1/\nu}$, and the real-space renormalization group cell-to-site fixed point estimator of Reynolds, Klein, and Stanley [54], $p_r(L)$, defined such that

$$R_L[p_r(L)] = p_r(L). \tag{6}$$

Numerical estimates for these quantities are shown in Table III.

The estimators p_m and p_{cc} are believed to approach their limiting values on the infinite lattice as $L^{-1-1/\nu}$, where $\nu = 4/3$ [33,34]. The estimator p_h is believed to converge to its limit at a faster rate of $L^{-1-\omega-1/\nu}$, where Ziff and Newman have determined a value of $\omega = 0.90(2)$ [34] for the scaling exponent proposed by Aharony and Hovi [55,56]. The estimator p_r is believed to approach its limit as $L^{-1/\nu}$, a much slower rate of convergence than for the other estimators [34]. Although each of these four limits is numerically equivalent to the percolation threshold p_c, it will be useful to adopt a general notation indicating the origin of any threshold estimates.

To second order, the finite size scaling relation for the median-p estimator is

$$p_m(L) \approx p_m^* - aL^{-1-1/\nu} + bL^{-1-\omega-1/\nu}. \tag{7}$$

A parametrized fit of Eq. (7) to the data of Table III produces $p_m^* = 0.592\,745\,95(4)$, $a = 0.413(5)$, and $b = 0.0(4)$. That the coefficient b is indistinguishable from zero suggests that a first-order model [Eq. (7) with b constrained to zero] is appropriate for the data. As such, a precise value for ω is unimportant. In this first-order case, the coefficients are evaluated as $p_m^* = 0.592\,745\,96(3)$, and $a = 0.4135(7)$. The very good agreement between model and experiment is shown in Fig. 2. An empirical power-law fit of the form $p_m(L) \sim p_m^* - aL^z$ yields $p_m^* = 0.592\,745\,95(5)$, $a = 0.42(2)$, and $z = -1.75(8)$. That the value of z is indistinguishable from the

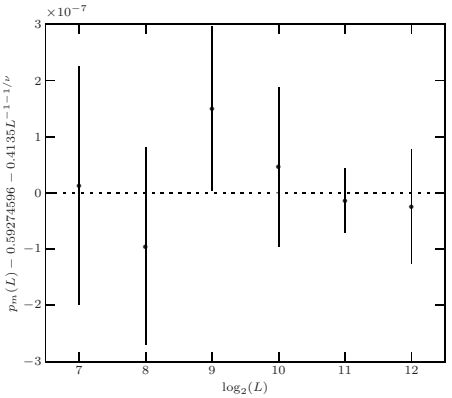

FIG. 2. Parametrized fit of first-order scaling theory [Eq. (7) with $b=0$] to experimental data [$p_m(L)$ of Table III] for the median-p critical point estimator.

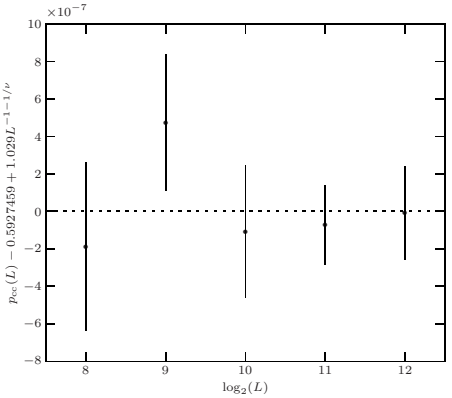

FIG. 3. Parametrized fit of first-order scaling theory [Eq. (8) with $c=0$] to experimental data [$p_{cc}(L)$ of Table III] for the cell-to-cell critical point estimator.

assumed exponent of $-1-1/\nu$, further supports scaling of the form $L^{-1-1/\nu}$ as being the appropriate model for the data at this level of precision. All three p_m^* estimates are in good agreement with one another.

The second-order scaling relation for the cell-to-cell estimator is given by

$$p_{cc}(L) \approx p_{cc}^* + \frac{a}{\alpha}L^{-1-1/\nu} + cL^{-1-\omega-1/\nu} \tag{8}$$

[34]. The quality of the cell-to-cell data is lower than that of the median-p data, as each point is obtained from the intercept of two lines, with statistical uncertainties, at a shallow angle, and as $p_{cc}(L)$ and $p_{cc}(L/2)$ are not entirely independent. A parametrized fit of Eq. (8) to the data of Table III produces $p_{cc}^*=0.592\,7458(2)$, $a=0.441(5)$, and $c=-9(8)$. Coefficient c is not inconsistent with zero, and a first-order model [Eq. (8) with c constrained to zero] does fit the data, as shown in Fig. 3, with coefficients of $p_{cc}^*=0.592\,7459(2)$ and $a=0.417(4)$, in good agreement with the median-p estimator results. An empirical power-law fit of the form $p_{cc}(L)\sim p_{cc}^*-(a/\alpha)L^z$ yields $p_{cc}^*=0.592\,7458(2)$, $a=0.34(8)$, and $z=-1.71(4)$, consistent with the assumed $L^{-1-1/\nu}$ scaling relation. All three p_{cc}^* estimates are consistent with each other and with the estimates for p_m^*, although the precision is significantly lower.

The linear combination estimator of Eq. (5) was constructed by Ziff and Newman [34] so as to cancel the first-order terms of Eqs. (3) and (4), leaving a faster approach to the percolation threshold,

$$p_h(L) \sim p_h^* + \frac{b+\alpha c}{1+\alpha}L^{-1-\omega-1/\nu} \tag{9}$$

(to first order). A parametrized fit of this expression to the data of Table III is shown in Fig. 4 and produces $p_h^*=0.592\,745\,96(7)$, and $(b+\alpha c)=0.3(9)$. The threshold result is in good agreement with those obtained from the median-p estimator data. The value of $(b+\alpha c)$ is also in agreement, although this is not saying much given the large uncertainties. That this value is essentially indistinguishable from zero is a reflection of the rapid rate of convergence of the $p_h(L)$ estimator with L, as suggested by Eq. (9), and the relative lack of precision in the $p_h(L)$ data. This is unsurprising given that no higher-order terms were apparent in either the $p_m(L)$ or $p_{cc}(L)$ data sets. As such, the data was inadequate to empirically test the assumed scaling exponent and is even consistent with $p_h(L)=\text{constant}=p_h^*$, for which fitting the weighted mean gives $p_h^*=0.592\,7460(1)$.

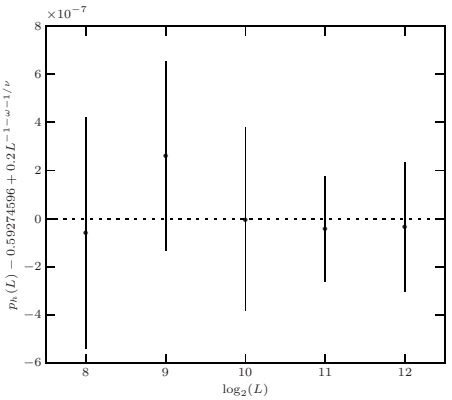

FIG. 4. Parametrized fit of scaling theory [Eq. (9)] to experimental data [$p_h(L)$ of Table III] for the linear combination critical point estimator.

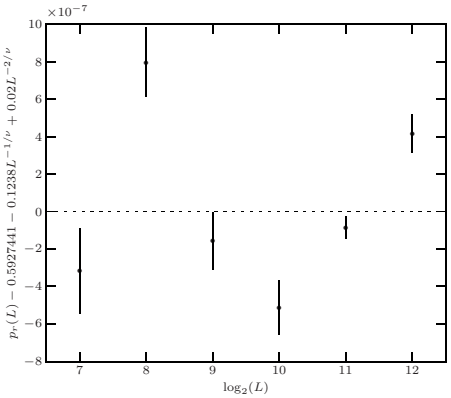

FIG. 5. Parametrized fit of scaling theory [Eq. (10)] to experimental data [$p_r(L)$ of Table III] for the renormalization group fixed point percolation threshold estimator.

The second-order scaling relation for the fixed point renormalization group estimator is given by

$$p_r(L) \sim p_r^* + rL^{-1/\nu} + sL^{-2/\nu} \tag{10}$$

[34], and so has a slower rate of convergence to its infinite lattice limit than any of the other estimators above. A fit to the data of Table III yields well-defined numerical values for the coefficients; $p_r^*=0.592\,7441(7)$, $r=0.1238(2)$, and $s=-0.021(6)$. However, as shown in Fig. 5, the model of Eq. (10) is but a loose match to the data at best, with higher-order terms evidently remaining significant. As such, the stated uncertainty in p_r^* is misleading and will be addressed within the next section. An empirical power-law fit of the form $p_r(L) \sim p_r^* - rL^z$ yields $p_r^*=0.592\,743(2)$, $r=0.1219(8)$, and $z=-0.748(2)$, consistent with the assumed first-order exponent of $-1/\nu$. The first-order model [Eq. (9) with s constrained to zero] returns $p_r^*=0.592\,7456(8)$ and $r=0.1233(1)$. Of course, neither of these two functions describe the data any better than does the second-order model.

VI. ROBUSTNESS

Several estimates have now been made for the square site percolation threshold, p_c, using all the data of Table III and with varying degrees of precision. Of these, the most precise is $p_c=p_m^*=0.592\,745\,96(3)$, obtained from the median-p estimator data using the first-order scaling model $p_m(L)=p_m^*+aL^{-1-1/\nu}$. It is prudent to establish the robustness of the results with respect to variations in the data and in the assumed model, over what domains the various models are valid, and how the domain and any fixed model parameters influence the estimate of p_c. There is a trade off between fitting to as much data over as great a domain as possible, so as to reduce statistical sampling fluctuations and hence to refine the result, and fitting to only data from large lattices where finite size effects are smaller and the scaling theories better describe the data. The results in Table I are consistent, where they overlap at $L=128$ and $L=256$, with those of Ziff and Newman [34]. Hence their data was used to extend the domain down to $L=8$ as necessary.

The fixed point renormalization group estimator is possessed of good quality data, but has a slow rate of convergence to its limit p_r^*. The $p_r(L)$ data of Table III can be reasonably well fit with the addition of an $L^{-3/\nu}$ term to the model, however the coefficient of $L^{-4/\nu}$, in an even higher-order model, is not zero. Values of the coefficients fluctuate with the order of the model, suggesting that even higher-order terms remain significant. Empirical power-law fits are consistent with the leading order exponent being $-1/\nu$, however a purely first-order model does not fit the data well until the domain is truncated to $L \geqslant 256$. Results for p_r^* are sensitive to the presence or absence of individual data points, the $L=4096$ point altering the result by $\pm 1\times 10^{-6}$. Under different models and data ranges, threshold estimates range from 0.592 744 to 0.592 746. The difference is much larger than the uncertainty in the individual estimates and so not much weight should be given to those. Consequently, although the raw data at a given L is relatively precise, the slow rate of convergence of the fixed point renormalization group estimator leads to only a very rough figure of $p_r^*=0.592\,745(1)$.

The linear combination estimator suffers from relatively large statistical uncertainties in the data, and points are not entirely independent of one another. However, the estimator does claim a very rapid rate of convergence to its limit, p_h^*. The model of Eq. (9) fits the data well for $L \geqslant 32$. Results thus obtained range from $p_h^*=0.592\,745\,94(5)$ to $p_h^*=0.592\,746\,03(8)$, with the presence or absence of individual data points making differences of as much as $\pm 4\times 10^{-8}$ in p_h^*. Allowing for alternative values of the parameter ω, between 0.85 and 0.95, the estimate changes by no more than $\pm 1\times 10^{-8}$. The data is not precise enough to either support or falsify the assumed scaling relation, and is not inconsistent with $p_h(L)=$constant. Even so, all estimates for p_h^* were consistent with one another and with the $128 \leqslant L \leqslant 4096 p_h(L)$ data mean of 0.592 7460(1). Hence the linear combination estimator appears to be robust, and the mean value, which covers the entire range of results, should be a more than safe estimate for p_h^*. The value of $p_h^*=0.592\,745\,96(7)$, obtained from all the $p_h(L)$ data of this study, should be reliable.

With similarly low data quality, nonindependent points, and a slower rate of convergence, the cell-to-cell renormalization group estimator should not be expected to provide any refinement in p_c over the linear combination approach. Over domains where the various models fit the data, cell-to-cell results for p_{cc}^* range from 0.592 7458(2) to 0.592 7461(2). Sensitivity to the presence or absence of individual data points is as for the linear combination results, but here this is much smaller than statistical uncertainties. The estimate $p_{cc}^*=0.592\,7459(2)$, obtained earlier from fitting the first-order scaling model to all the p_{cc} data of Table III, is in agreement with the entire range of cell-to-cell results above, and so is robust, if imprecise.

TABLE IV. Infinite lattice limit estimates for the percolation threshold. Results are shown, by estimator, for the Mersenne twister only data (MT, MTH, DMT), and also for the combined generators data (MT, MTH, DMT, SWB, QTA, QTB, XG).

Limit	Mersenne	Combined
p_r^*	0.592 745(1)	0.592 745(1)
p_{cc}^*	0.592 745 9(2)	0.592 745 9(2)
p_h^*	0.592 745 96(7)	0.592 745 98(6)
p_m^*	0.592 745 96(4)	0.592 745 98(3)

The median-p based estimates have the same rate of convergence as the cell-to-cell estimates, but with independent data points of much higher quality. The median-p estimates are less sensitive to the presence or absence of any one particular data point, this making a difference of at most 2×10^{-8}, and typically of less than 1×10^{-8}, in the result for p_m^*. The first-order model fits the data for $L\geqslant128$, with results lying in the range $p_m^*=0.592\,745\,94(3)$ to $p_m^*=0.592\,745\,96(4)$. The second-order model fits the data for $L\geqslant16$, with results lying between $p_m^*=0.592\,745\,91(8)$ and $p_m^*=0.592\,746\,00(5)$. The empirical power-law model makes a good fit for $L\geqslant64$, with estimates of p_m^* running from 0.592 745 89(2) up to 0.592 746 03(2), and scaling exponents in the range $-1.729(7)$ to $-1.79(2)$. As noted in the preceding section, the first-order fit matches the data of Table III very well, the empirical fit agrees with the assumed exponent of $-1-1/\nu$, and coefficients of higher-order terms were insignificant. This indicates that the first-order model does indeed provide an accurate description for the finite-size scaling behavior of the median-p estimator. The estimate thus obtained, of $p_m^*=0.592\,745\,96(3)$, does not quite encompass the entire range of results above. Allowing for an extreme scenario, where even the model and scaling exponent may not be quite right, a more conservative figure of $p_m^*=0.592\,745\,96(4)$ does cover all of the above results. Hence this final value of the median-p estimate for p_c should be quite dependable. Incidentally, a standard error of 4×10^{-8} in p_c is approximately what would be expected from the total amount of data sampled in this study [as listed in the $p_m(L)$ column of Table III]. Parameter a of Eq. (7) shows much more sensitivity to the data domain and model than does p_m^*. The fitted value given in the preceding section was the most precise obtained. An overall result of $a=0.415(5)$ is more reasonable in light of the other estimates.

The four estimators have now produced equally many robust estimates for the two-dimensional square site percolation threshold p_c. As summarized in Table IV, these are $p_r^*=0.592\,745(1)$, $p_{cc}^*=0.592\,7459(2)$, $p_h^*=0.592\,745\,96(7)$, and $p_m^*=0.592\,745\,96(4)$, in good mutual agreement. Taking $p_c=0.592\,745\,96$, and returning to the canonical spanning probability curves, a good match between the data of $128\leqslant L\leqslant4096$ and the theory of $R_L(p_c)\approx0.5+kL^{-1}+O(L^{-2})$ was had for $k=0.317(1)$. No higher-order terms were seen, with the coefficient of L^{-2} being indistinguishable from zero. The value of k found here is a little lower than those of Ziff, $k=0.319(1)$ [33], and Newman and Ziff, $k=0.320(1)$ [29]. The difference in $p_f(2048)$ resulting from using $k=0.317$, as opposed to $k=0.320$, in Eq. (2) is around 6×10^{-9}. This is much less than the statistical uncertainties in the results of Table I, upholding the claimed insensitivity of those estimates to k. Hence those results remain reasonable (PRNG biased) estimates for p_c, and the direct comparison with earlier p_c estimates is valid. The estimate $a=0.415(5)$ found above is consistent with the results of Ziff, Newman, Hovi, and Aharony [33,34,56] (a here equates to their ratio b_0/a_1). Since k equates to b_0, it follows that $a_1=0.76(1)$ from the data obtained within this exercise. This estimate is also consistent with those of previous works [33,34,56].

The above results are based on data acquired solely from the Mersenne twister, that generator having been determined as suitable for this problem. In Sec. IV, results obtained from the SWB, QTA, QTB, and XG generators were found to be consistent with results obtained from the Mersenne twister. Although those four generators were not tested to the same extent as Mersenne twister, there is no objective reason to discount them entirely. Incorporating the data obtained from these generators earlier leads to revised values of $p_m(2048)=0.592\,745\,32(4)$, $p_r(2048)=0.593\,150\,55(4)$, $p_{cc}(2048)=0.592\,7476(1)$, $p_{cc}(4096)=0.592\,7463(2)$, $p_h(2048)=0.592\,7459(1)$, and $p_h(4096)=0.592\,7459(3)$ for the various estimators of Table III. Note that the majority of the data remains Mersenne twister based.

Use of these revised values does not alter either the fixed point limit, p_r^*, or the cell-to-cell limit, p_{cc}^*. The linear combination limit is raised to $p_h^*=0.592\,745\,98(6)$, an adjustment of rather less than its statistical uncertainty.

A parametrized fit of Eq. (7) to the revised median-p data yields $p_m^*=0.592\,745\,98(3)$, $a=0.415(7)$, and $b=0.1(5)$. As before, the coefficient of the higher-order term is indistinguishable from zero. A first-order fit [of Eq. (7) with b constrained to zero] produces $p_m^*=0.592\,745\,98(3)$, and $a=0.414(1)$. An empirical power-law fit of the form $p_m(L)\sim p_m^*-aL^z$ finds $p_m^*=0.592\,745\,98(4)$, $a=0.41(2)$, and $z=-1.75(1)$. The excellent agreement between this model and the experimental data is shown in Fig. 6. The fitted value of z is indistinguishable from the assumed scaling exponent of $-1-1/\nu$ (with $\nu=4/3$). All fitted parameters are consistent across the three models. Note that if only the QTB and XG data were combined with that from the Mersenne twister (the SWB and QTA generators being suspect on general grounds [48,49], although no obvious problems were seen here), then the revised median-p estimator would be $p_m(2048)=0.592\,745\,31(5)$, leading to $p_m^*=0.592\,745\,98(3)$ from the first-order fit and $p_m^*=0.592\,745\,98(4)$ from the empirical fit. These two values are identical to those obtained with the inclusion of SWB and QTA based data.

Performing robustness checks as before, the first-order model fits the data for $L\geqslant128$, with results lying within a worst case range of $p_m^*=0.592\,745\,98(6)$ to $p_m^*=0.592\,745\,99(8)$, and much more typically within $p_m^*=0.592\,745\,98(2)$ to $p_m^*=0.592\,745\,99(4)$. The second-order model fits the data for $L\geqslant16$, with results lying between $p_m^*=0.592\,745\,96(3)$ and $p_m^*=0.592\,746\,02(4)$. The empirical power-law model makes a good fit for $L\geqslant128$, with estimates of p_m^* running from 0.592 745 98(4) up to

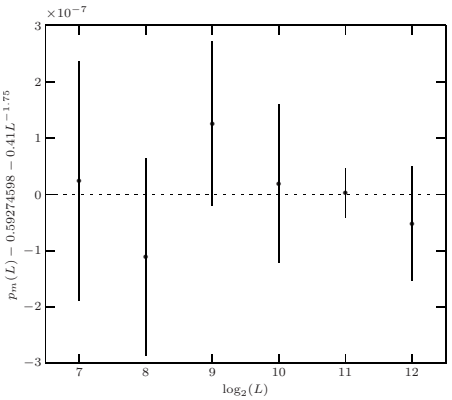

FIG. 6. Parametrized fit of empirical power-law model $p_m(L) = p_m^* - aL^z$ to combined experimental data from the Mersenne twister, subtract with borrow, generalized xorshift, and both quad-tap generators.

0.592 745 99(8), and scaling exponents, z, in the range $-1.74(1)$ to $-1.77(1)$. Hence, the data supports the validity of the first-order model with the assumed scaling exponent, and a standard error of 3×10^{-8} in p_m^* appears fully justified. The various threshold estimates are summarized in Table IV. Using the revised data, and $p_c = 0.592\ 745\ 98$, the estimate of the finite size correction parameter remains unchanged at $k = 0.317(1)$. Nor is any significant change seen in parameter a.

Assuming the suitability of the Mersenne twister PRNG for this particular Monte Carlo application, and also assuming that the median-p estimator approaches the critical point as $p_m(L) - p_c \propto L^{-1-1/\nu}$, where $\nu = 4/3$, as supported by the data, then a robust estimate for the square site percolation threshold is $p_c = 0.592\ 745\ 96(4)$. A value for ω is not required. Further assuming the suitability of the generalized xorshift and QTB generators, the additional data adjusts this estimate to $p_c = 0.592\ 745\ 98(3)$ (this value does not change if the subtract with borrow and QTA generators are additionally assumed to be suitable). Continuing to assume the reliability of those generators, while dropping the assumed scaling exponent and requiring only that $p_m(L) - p_c \propto L^z$, for some z, the estimate becomes $p_c = 0.592\ 745\ 98(4)$. These three estimates are mutually consistent to well within statistical uncertainties. The most precise of them has a standard error of 3×10^{-8}, however a degree of caution is warranted in that none of the generators were tested to that level of precision. That being the case, this study's final estimate for the square site percolation threshold is

$$p_c = 0.592\ 745\ 98(4). \tag{11}$$

This, primarily Mersenne twister based, estimate is consistent with almost all previous results in Table II. In particular, it is in good agreement with the Mersenne twister derived estimate of Lee. Taken collectively however, those results, excluding that of Lee, would suggest a higher value for p_c, in the vicinity of 0.592 7463(1). Although the value obtained here lies well outside of that range, the difference could be attributable to the various pseudo-random-number generators used. While it is not impossible that the result obtained here may reflect some detectable influence of the chosen generators, precautions against this were taken and no evidence of bias was found.

VII. CONCLUSIONS

Increasing availability of highly parallel computer facilities now makes it practical to obtain significant quantities of Monte Carlo data from large lattices. This allows for greater precision in derived statistics, but requires very good quality pseudo-random-number generators as it is well established that inadequate generators lead to erroneous results.

Tests were performed upon several generators and it was found that use of simple two-tap generators should probably be avoided for this application. The MT19937 generator appeared to be suitable and was adopted for the majority of the Monte Carlo sampling conducted within this study. No dependence was found upon the (reasonable) choice of generator initialization.

Percolation threshold estimates subsequently made from various crossing probability statistics were found to be in good mutual agreement. The most precise of these was obtained from the median-p estimator. Data quality was such that precise results could be obtained without the need to assume a particular scaling exponent. Even so, results were in good agreement with a leading exponent of $-1-1/\nu$ and no higher-order term was found. The square site percolation threshold was subsequently determined to be $p_c = 0.592\ 745\ 98(4)$.

This estimate is consistent with the majority of earlier results on an individual basis, but not with those same results combined. Evidence suggests, however, that at least some of those earlier results have been influenced by the pseudo-random-number generators used. The generators used here appear to be of adequate quality, and the main generator, MT19937, passed an application specific test of randomness. Furthermore, efforts were made to ensure the reliability of the error bounds in that final estimate, which should, then, be accurate.

ACKNOWLEDGMENTS

The author would like to thank R. M. Ziff for raising the need to address the percolation problem with different generators, for helpful advice and discussions, and for comments on the paper. Thanks are also extended to the referees for their comments, to H. W. J. Blöte, R. P. Brent, and P. H. L. Martins for providing details on their respective pseudo-random-number generators, and to A. J. E. Dale, G. L. Evans, and C. J. McMurtrie for assistance with the University of Canterbury's IBM Blue Gene supercomputing facility, upon which the Monte Carlo sampling was conducted.

[1] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, 2nd ed. (Taylor and Francis, London, 1994).
[2] B. Bollobás and O. Riordan, *Percolation* (Cambridge University Press, Cambridge, 2006).
[3] M. Sahimi, *Applications of Percolation Theory* (Taylor and Francis, New York, 1994).
[4] T. Harris, Proc. Cambridge Philos. Soc. **56**, 13 (1960).
[5] H. Kesten, Commun. Math. Phys. **74**, 41 (1980).
[6] M. Sykes and J. Essam, Phys. Rev. Lett. **10**, 3 (1963).
[7] M. Sykes and J. Essam, J. Math. Phys. **5**, 1117 (1964).
[8] J. C. Wierman, J. Phys. A **17**, 1525 (1984).
[9] R. M. Ziff and C. R. Scullard, J. Phys. A **39**, 15083 (2006).
[10] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 2nd ed. (Addison-Wesley, Reading, MA, 1981).
[11] D. Lehmer, *Proceedings of the 2nd Symposium Large-Scale Digital Calculating Machines* (Harvard Univeristy Press, Cambridge, MA, 1951), pp. 141–146.
[12] G. Marsaglia, in *Applications of Number Theory to Numerical Analysis*, edited by S. K. Zaremba (Academic, New York, 1972), pp. 249–285.
[13] R. C. Tausworthe, Math. Comput. **19**, 201 (1965).
[14] T. G. Lewis and W. H. Payne, J. ACM **20**, 456 (1973).
[15] S. Kirkpatrick and E. P. Stoll, J. Comput. Phys. **40**, 517 (1981).
[16] G. Marsaglia and A. Zaman, Ann. Appl. Probab. **1**, 462 (1991).
[17] R. M. Ziff, Comput. Phys. **12**, 385 (1998).
[18] M. Matsumoto and T. Nishimura, ACM Trans. Model. Comput. Simul. **8**, 3 (1998).
[19] P. L'Ecuyer, Math. Comput. **68**, 249 (1999).
[20] G. Marsaglia, Random numbers for C: End, at last, Sci. Stat. Math Web discussion, 1999.
[21] G. Marsaglia, J. Stat. Software **8**, 1 (2003).
[22] W. W. Tsang, L. C. K. Hui, K. P. Chow, C. F. Chong, and C. W. Tso, Proceedings of the 27th Australian Conference on Computer Science, 2004, Vol. 26, p. 23.
[23] R. P. Brent, ANZIAM J. **48**, C188 (2007).
[24] P. L'Ecuyer and R. Simard, ACM Trans. Math. Softw. **33**, 22 (2007).
[25] A. Kolmogorov, Russ. Math. Surveys **38**, 29 (1983).
[26] G. Chaitin, *Information, Randomness and Incompleteness* (World Scientific, Singapore, 1987).
[27] A. Compagner and A. Hoogland, J. Comput. Phys. **71**, 391 (1987).
[28] M. E. J. Newman and R. M. Ziff, Phys. Rev. Lett. **85**, 4104 (2000).
[29] M. E. J. Newman and R. M. Ziff, Phys. Rev. E **64**, 016706 (2001).
[30] Y. Deng and H. W. J. Blöte, Phys. Rev. E **72**, 016126 (2005).
[31] H. W. J. Blöte (private communication).
[32] R. M. Ziff and G. Stell, Laboratory for Scientific Computing, University of Michigan. Report No., 88-4, 1988, footnote 26.
[33] R. M. Ziff, Phys. Rev. Lett. **69**, 2670 (1992).
[34] R. M. Ziff and M. E. J. Newman, Phys. Rev. E **66**, 016129 (2002).
[35] P. Balister, B. Bollobás, and M. Walters, Random Struct. Algorithms **26**, 392 (2005).
[36] O. Riordan and M. Walters, Phys. Rev. E **76**, 011110 (2007).
[37] M. J. Lee, Phys. Rev. E **76**, 027702 (2007).
[38] A. Compagner, Phys. Rev. E **52**, 5634 (1995).
[39] J. Hoshen and R. Kopelman, Phys. Rev. B **14**, 3438 (1976).
[40] R. M. Ziff (private communication).
[41] R. M. Ziff and B. Sapoval, J. Phys. A **19**, L1169 (1986).
[42] F. Yonezawa, S. Sakamoto, and M. Hori, Phys. Rev. B **40**, 636 (1989).
[43] C.-K. Hu, Chin. J. Phys. (Taipei) **32**, 519 (1994).
[44] C.-K. Hu, Phys. Rev. B **51**, 3922 (1995).
[45] C.-K. Hu, C.-N. Chen, and F. Y. Wu, J. Stat. Phys. **82**, 1199 (1996).
[46] P. H. L. Martins and J. A. Plascak, Phys. Rev. E **67**, 046119 (2003).
[47] P. H. L. Martins (private communication).
[48] M. Matsumoto and Y. Kurita, ACM Trans. Model. Comput. Simul. **6**, 99 (1996).
[49] S. Tezuka, P. L'Ecuyer, and R. Couture, ACM Trans. Model. Comput. Simul. **3**, 315 (1993).
[50] F. Panneton, P. L'Ecuyer, and M. Matsumoto, ACM Trans. Math. Softw. **32**, 1 (2006).
[51] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Phys. Rev. Lett. **69**, 3382 (1992).
[52] P. Reynolds, H. Stanley, and W. Klein, J. Phys. A **11**, L199 (1978).
[53] P. J. Reynolds, H. E. Stanley, and W. Klein, Phys. Rev. B **21**, 1223 (1980).
[54] P. J. Reynolds, W. Klein, and H. E. Stanley, J. Phys. C **10**, L167 (1977).
[55] A. Aharony and J.-P. Hovi, Phys. Rev. Lett. **72**, 1941 (1994).
[56] J.-P. Hovi and A. Aharony, Phys. Rev. E **53**, 235 (1996).

Chapter 4

Further Applications

This chapter discusses some further applications of the algorithms introduced earlier. These include studies of boundary effects (cluster density mapping), site-bond percolation models, and transport phenomena.

4.1 Transport Phenomena

As noted at the beginning of chapter 1, initial efforts were directed toward the development of a stochastic cellular automaton for simulating exciton dynamics on a surface. An exciton is a high energy, high angular momentum, atomic state of a host ion that is itself embedded within some inert matrix. Excitons may exist only upon the host sites (not within the surrounding material) and each such site may host a maximum of one exciton. This exciton concept was developed by Frenkel [74], and later given a transport mechanism by Förster [71] and Portis [204]. Due to its high angular momentum, the presence of an exciton results in a significant magnetic dipole-dipole interaction between the host site and any nearby sites. This perturbation may cause another exciton to form on one of those sites, and energy conservation results in the original exciton decaying into the ground state. Consequently, a single exciton will appear to travel through the medium via a series of discrete hops [215]. The probability of making such a hop is determined by the strength of the dipole-dipole interactions, and is based upon the Bloch equations [20]. This probabilistic hopping nature is what provoked the automaton approach [226, 188, 279]. Such predictive models can be

tested against observational data from photon echo experiments [161, 162]. In these, a laser is used to excite the host sites within some localised region, from which the excitons proceed to disperse naturally.

The exciton transport model supports substitutional disorder and diagonal disorder [217]. Substitutional disorder refers to irregularity in the spatial distribution of host sites. A material has substitutional disorder if not all possible host sites are occupied. Diagonal disorder refers to adjustments to the hopping probability, resulting from perturbations to the Hamiltonian of each host site, caused by spatial variations in the crystal field and/or some applied external field [151, 238]. A material has diagonal disorder when not all host sites have the same excitation energy. When the variations in energy are spatially uncorrelated, diagonal disorder is also referred to as inhomogeneous broadening since the width of the observed spectral line increases as the exciton spatial distribution widens from diffusion.

4.1.1 Discrete Exciton Models

Our first attempt at a viable model was constructed around the Hamiltonian of Root and Skinner [216]. This consists of three terms; an unperturbed (constant) host site excitation energy, a diagonal disorder perturbation to that energy (caused by local irregularities in crystal structure, which are, in part, due to substitutional disorder of the host sites), and an exciton-exciton interaction energy (due to the repulsive dipole-dipole Förster mechanism) [215]. The Hamiltonian does not, however, fully specify the simulational model dynamics.

In the first model, substitutional disorder was low and the dynamics consisted of considering each exciton in turn, randomly choosing a vacant neighbour site (if one or more existed), and moving the exciton between sites with a probability calculated from the energy difference between the initial and final states (in the manner of the Metropolis [179] or Barker algorithms [12]). The excitons slowly drift apart under the influence of Fermi pressure and the dipole-dipole interaction. By imposing a gradient in the diagonal disorder (this was to have been a feature of the photon echo experiments), the rate and direction of exciton drift could be controlled. However, in order to conserve energy in this non-thermodynamic system, a demon algorithm had to be employed [44]. A side effect of this was that the number of excitons tends to increase with time and this act as a pseudoforce driving the exci-

tons toward lower energy hosts. So, although the model loosely exhibits the expected behaviour, there was too much guess work in the details regarding the all important drift rates. There are also issues concerning the interaction range; excitons will most likely move nearest neighbour distances (the Förster hopping probability falls off as $1/r^6$), but when no nearest neighbour exists, they will eventually hop to the next nearest neighbour.

A second model was based on ringing, rather than hopping, dynamics. For each site in the system, the time dependent perturbation (resulting from exciton angular momentum) can be calculated by summing over all excitons present. Then, each site makes a transition between the unexcited and excited states (or *vice versa*) with a well defined probability. The trouble here, is that in order to conserve energy, these transition events must be entangled, and, with multiple excitons in the system, it is far from clear as to which exciton should be deactivated whenever an inactive site activates (failure to do this results in a runaway exciton count).

A third model was a compromise between the first two, and run upon a highly disordered lattice. The neighbours of each site were considered to be those other sites to which the site was adjacent in the Delaunay triangulation [273]. This way excitons will move the shortest distances they can, but will still attempt to move when those shortest distances are relatively large. Then, for each exciton in the system, a single neighbouring site was chosen at random, and, provided that site was vacant, the exciton would hop there with a probability determined as in the ringing model. Again, this model ran into problems whenever multiple excitons were in close proximity, and the underlying problem seemed to be the attempt to treat the strongly interacting excitons as discrete individual particles.

4.1.2 Limits of the Exciton

As it turns out, these difficulties (and more) had already attracted considerable attention within the research community. Shortly after Broadbent and Hammersley put percolation theory into its current form [25], Anderson produced a (now famous) paper on exciton diffusion on random lattices [5]. In this paper, Anderson considered a model with diagonal disorder, but not substitutional disorder. At low (non-interacting) exciton densities, transport was governed by the Förster mechanism (single real hops). At high exciton densities, transport was governed by probability wavepacket

extension (multiple virtual hops). The low density case also describes spin diffusion and is known to show a percolative phase transition [193]. The high density case has become known as the Anderson model. The Anderson, or delocalisation, transition refers to a phase transition where the wave packets extend throughout the entire lattice.

The analogy of the delocalisation transition to percolation theory is obvious, and, in 1972, Kirkpatrick and Eggarter laid the foundations of quantum percolation theory in a paper concerned with localisation and the Förster mechanism [135]. Quantum percolation was defined as a second stochastic (bond conductance) layer over top of the classical percolation model [194, 233]. Bonds have to exist in the classical model before they can be tested for conductance, and so classical percolation was a prerequisite to quantum percolation, the latter necessarily having a higher threshold value. The delocalisation transition was considered to have taken place when the spanning quantum percolation cluster had come into existence. In one of the ten most cited letters to Physical Review, Abrahams, Anderson, Licciardello and Ramakrishnan showed that no delocalisation transition takes place in the two dimensional Anderson model [1]. Frölich and Spencer then proved a non-transport theorem for systems of high disorder (the Green's function of the Anderson Hamiltonian decays exponentially fast) [77]. Under such conditions, the nature of the lattice may even become blurred to the point where the location of sites might better be described in terms of a random graph [27, 198], in which case what constitutes a neighbouring site is not so clearly defined, since the magnetic interactions have long range (although adjacent sites in the Delaunay triangulation [273] are one possibility). In such cases, the timescale of the experiment becomes important [71, 215, 238]. Excitons will eventually move, but perhaps not for quite some time [98, 223]. The problem is made complicated by the fermionic nature of excitons. Each site may host at most a single exciton, and so the problem is more difficult than a simple probability amplitude flow upon a mesh.

However, most quantum percolation models were concerned with substitutional disorder and off-diagonal disorder (where the bond conductance probability is independent of the adjacent sites), rather than with diagonal disorder (where the bond conductance probability does depend upon the sites) [63]. The two models were eventually combined [217], and Taylor and MacKinnon then showed that a delocalisation transition is also absent

from the appropriate two dimensional quantum percolation model [259]. At around the same time, the Bloch equations failed experimental tests [52, 232] and were subsequently generalised [16]. However, the concept of the individual exciton, at least at high densities, seemed flawed [103, 128]. In another of the ten most cited letters to Physical Review, Ceperley and Alder had already shown that collective motion could be important [30], and experimental evidence that this was indeed the case would be forthcoming [180] (see also [34]).

To make a long story short, attempts to describe exciton transport phenomena dynamically are fraught with difficulty (it is easy enough to construct a mathematical model, but these may not have much to do with physical reality). In particular, whenever multiple excitons are present in the system the very concept of the discrete exciton breaks down, and any quasiparticle model, cellular automata included, will encounter problems. However, there are conditions upon the medium that need to satisfied before transport can take place, and these can be described in terms of a quantum or site-bond percolation model [33, 235, 183, 142].

4.1.3 Site-Bond Percolation

As a means of locating the critical surface of the exciton transport phase diagram, the algorithms of chapter 2 were initially developed with the site-bond problem in mind. On a square lattice of $N = L \times L$ sites, there are of course $2L(L-1)$ bonds. In a classical site percolation model, all bonds can be considered occupied with sites only probabilistically occupied. In a classical bond percolation model, all sites are occupied while bonds are probabilistically occupied. In a site-bond model, both sites and bonds are probabilistically occupied [33, 246], usually independently of one another, and a cluster consists of an irreducible set of occupied sites pathways connected by occupied bonds (see section 3.1).

An example is shown in figure 4.1, where the algorithms of chapter 2 confer the ability to move over the microcanoncial spanning probability surface in arbitrary directions. It is worth noting that, so far as the spanning properties of a given lattice configuration are concerned, there is no difference between a site being unoccupied and all the bonds about it being unoccupied. It follows that the site model can be implemented by unoccupying all the bonds adjoining a site rather than by unoccupying the site itself.

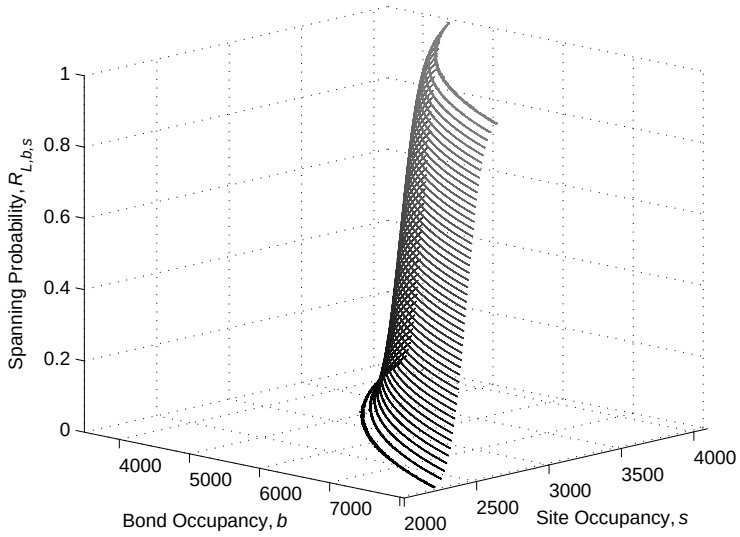

Figure 4.1: Hybrid site-bond percolation model spanning probability surface, $R_{L,b,s}$, showing the probability for the existence of a horizontally spanning cluster, on an $L = 64$ square lattice in the microcanoncial ensemble, with s sites occupied and b bonds occupied. Contour lines run from $R_{L,b,s} = 0.01$, up to $R_{L,b,s} = 0.99$, at intervals of 0.02.

Consequently the only real difference between the site and bond percolation models is of spatial correlation (or lack thereof) in the bonds removed.

4.1.4 Transport Phase Diagram

Figure 4.2 shows the transport phase diagram for two dimensional square lattices with both substitutional (site) disorder and off-diagonal (bond) disorder (the figure is an accurate depiction of a renormalisation group schematic due to Nakanishi and Reynolds [186, 246]). This is derived from figure 4.1, and applies to physics where the excitons cannot cross gaps of more than one lattice spacing but will otherwise move everywhere that is accessible to them. Although no information is provided regarding the speed of exciton diffusion, the final extent (dispersed or localised) is clearly indicated. This itself is experimentally testable, and the method used in obtaining the figure is easily modified to include diagonal disorder.

Naturally, this is model is a little simpler than the physics, since, in

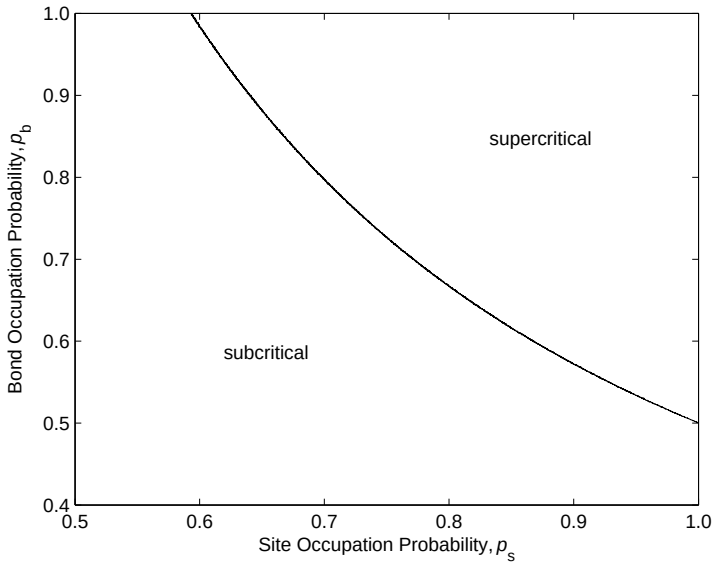

Figure 4.2: Transport phase diagram for a square lattice with substitutional disorder, without diagonal disorder, and with a quantum (stochastic) component to the formation of transporting (conducting) bonds. This is a equivalent to a hybrid site-bond percolation model with sites occupied at probability p_s and bonds occupied with probability p_b. The boundary line, between the supercritical (long range transporting) and subcritical (localised, non-transporting) phases, terminates at the classical site and bond percolation thresholds.

practice, excitons perturb the Hamiltonian of nearby hosts [182]. This phenomena is known as excitation-induced frequency shift. Fortunately, the shift is quite small (typically much less than 1 MHz) [122, 158], and so figure 4.2 should be visually accurate at least.

4.2 Finite Size Effects

Boundary effects are of perpetual importance in percolation, where results from Monte Carlo calculations, necessarily conducted upon finite sized lattices, must be extrapolated to the infinite sized system where the phase transition actually occurs [160, 196, 246, 290]. In some models, asymptotic limits can differ from known quantities by as much as one percent, even if finite lattices of truly enormous size were to be used [96]. Clearly the very

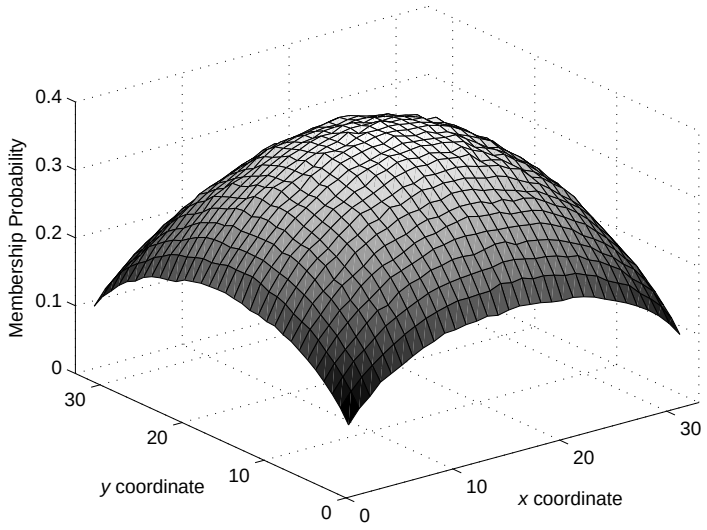

Figure 4.3: Map of the probability that any given site on the $L = 32$ square lattice is a member of the incipient spanning cluster in the microcanonical ensemble. Finite size boundary effects are manifest in the non-zero curvature of this surface.

existence of a boundary makes qualitative differences to the system, and an understanding of these is required.

4.2.1 The Incipient Spanning Cluster

As described in subsection 2.5.3, a straightforward addition to the basic data structures and algorithms of chapter 2 permits rapid identification of all member vertices of a chosen connected graph without any significant reduction in performance. This ability can be used to investigate boundary effects upon spanning clusters on finite sized lattices (see chapter 3). Beginning with an empty lattice, sites are randomly occupied until a spanning cluster exists, at which point the member sites of that cluster are identified and recorded. Repeating this procedure many times, a map of the spanning cluster membership probability distribution for lattice sites at the onset of percolation can be built up (that is, for the incipient spanning cluster [242, 236]). The result of such an exercise can be seen in figure 4.3.

The shape of this surface (figure 4.3) changes very little with lattice size.

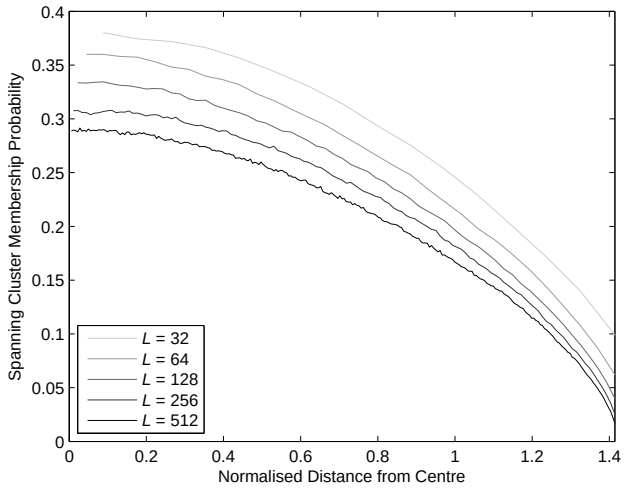

Figure 4.4: Cross section through the map of figure 4.3, running outward from the centre along straight lines to the lattice corners. Results are shown for lattices of up to $L = 512$. The horizontal axis' unit of distance is L, with the Euclidean metric used. Observe that boundary effects remain significant even as the lattice becomes large.

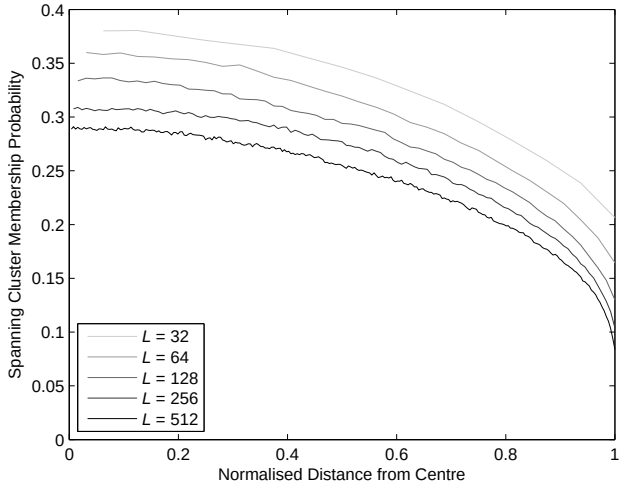

Figure 4.5: Cross section through the map of figure 4.3, running outward from the centre along straight lines perpendicular to the lattice boundary. Results are shown for lattices of up to $L = 512$. The horizontal axis' unit of distance is L. Observe that boundary effects are less pronounced than along the diagonal cross section of figure 4.4.

This is illustrated by the cross sections of figures 4.4 and 4.5. Empirically, the data shows that the spanning cluster membership probability drops off roughly as a power, z, of the normalised distance from the centre (provided that this distance is not too close to the boundary). Along the diagonal cross section (figure 4.4) $z \approx 1.9$, and along the straight cross section (figure 4.5) $z \approx 2.0$. Evidently, the reduction in membership probability at the boundary is actually more pronounced in larger finite sized lattices. Finite size effects clearly remain important even for very large lattices, with boundary influences penetrating deep into the lattice.

4.2.2 Distribution and Correlation Tests

Data of this sort is often used in tests of analytic results derived from Cardy's formula, Schramm-Loewner evolution, or other methods. These have included experimental tests of exact results for cluster area distributions [295, 29] and two-point correlation functions [60, 126, 137, 237], amongst others. The connection to figure 4.3 is obvious, and so the algorithms could well be of use in these endeavours.

4.3 Future Possibilities

This section lists some possible percolation related applications and modifications of the algorithms introduced in chapter 2. None of these were investigated in any real depth, however they might provide interesting lines of future study.

4.3.1 Other Topologies

Chapter 3 was concerned with a single specific model; site percolation on the two dimensional square lattice. There are, of course, similar problems on other topologies in other dimensions [249, 199]. A small list of these appears in table 3.1. It would be interesting to see how the algorithms of chapter 2 fare with these models, as the topology will influence the computational performance of the edge removal algorithm.

4.3.2 Importance Sampling

The results of section 4.2 raise an intriguing possibility that the range sweeping method of chapter 3 might be combined with an importance sampling technique [88]. Rather than choosing sites from the lattice with uniform probability, they could instead be chosen with a probability reflecting figure 4.3. Such an approach might decrease the effective correlation time in the sequence of Monte Carlo samples, thereby improving the overall performance of the method.

4.3.3 Fractal Mass Dimension

Since the algorithms of chapter 2 must keep track of the mass of each cluster as a matter of course, it becomes almost trivial to investigate such things as the mass distribution function for clusters. Of particular interest might be the fractal mass dimension for the spanning cluster at the critical point [166, 242]. On a finite lattice of N sites at the percolation threshold (in the canonical ensemble), the mean mass of the spanning cluster scales as N^z for some non-integer z [245]. The algorithms are potentially well suited to the determination of this parameter.

4.3.4 Excess Cluster Numbers

As described in subsection 2.5.2, and applied in chapter 3, the sweeping method is easily able to track the distribution of arbitrary cluster properties. Among these are lattice spanning properties, making short work of establishing the number of non-spanning (excess) clusters present on the lattice at any given time. The distribution of the number of excess clusters is of interest [187, 245, 246] and the sweeping method is able to take the necessary measurements, while primarily engaged in spanning probability measurements as in chapter 3, with almost no additional computer time required.

4.3.5 Virtual Algorithm Steps

Consider the exercise of chapter 3 where the only observational quantity was the existence or otherwise of a lattice spanning cluster. The spanning properties method of subsection 2.5.1 is able to determine the number of

clusters with each observable spanning property, as a result of occupying (or deoccupying) another lattice site, in slightly less computer time than it takes to actually perform the site occupation or deoccupation by the algorithms of section 2.4. Consequently, if enormous amounts of computer time are available, it may prove advantageous to consider all possible lattice configurations one Monte Carlo step removed from the current configuration. That is (for the sake of argument, assume that the next Monte Carlo step is to occupy a site), from the current lattice configuration, calculate and record the spanning properties data that would result from the occupation of all unoccupied sites in turn (these are the virtual steps) before choosing one site at random and actually performing the occupation procedure (this is the real Monte Carlo step). In this way, improved estimates of the spanning cluster existence probability can be obtained. However, for a lattice of any reasonably large size, very large amounts of computer time will be required to perform all the virtual steps.

4.3.6 Directed Percolation

The graph structures of subsection 2.3.1 represent edges with pointers between vertex objects. These pointers are directed, from one object to another, and so undirected edges are really pairs of pointers, one from each of the objects to the other. It is not strictly necessary to have both pointers, or to limit the number of pointers from one object to another to, at most, one. Hence the algorithms of chapter 2 are tailor made for directed percolation models and multiple bond models [134, 93, 268, 201].

4.3.7 Approximate Method

This subsection describes an approximate method based on (loosely speaking) real space cell-to-cell renormalisation [267, 211, 292]. The method did not work well enough to justify a more thorough investigation. To begin, spanning clusters were formed upon a large number of 32×32 square site lattices. For each of these lattices, the sites along each boundary that are members of the spanning cluster were stored as a 32 bit pattern of a single computer word. A fifth word recorded the number of sites on that lattice. Collections of randomly chosen and oriented cells then formed the sites of a much larger lattice. In this way much larger lattices could be considered,

as only five words are required in memory for each cell (only the boundaries being important). Bonds were considered to exist between cells when the spanning cluster boundary sites were adjacent (that is, when the logical and of the appropriate boundary words from the two cells was not zero). The existence or otherwise of a spanning cluster could then be determined for the entire lattice of cells, and, since the total number of occupied sites was known, this could be used to estimate the percolation threshold. In practice, it was found that a spanning cluster only existed over the entire lattice of cells when almost all sites on the cells were occupied (so that a reasonable number of boundary sites belonged to the spanning cluster), and so this led to a huge overestimate of the threshold. This is not surprising in light of the results of section 4.2, however, it remains possible that some variation on this approach could be useful.

4.3.8 Bootstrap Percolation

An interesting problem that the algorithms may be well suited to is bootstrap percolation [279, 94, 95]. In this model, a cellular automaton is allowed to evolve in such a way that the number of live cells increases with time, and eventually an incipient spanning cluster will exist. The algorithms of chapter 2 make it easier to consider automata where live cells often die, since the algorithms enable rapid recalculation of spanning properties on vertex removal (cell death). For the very same reason, the algorithms may also enable an improvement in the binary search method of subsection 3.3.2.

4.4 Summary

This chapter has demonstrated the application of the algorithms from chapter 2 to a site-bond percolation model, to excitonic transport problems, and to the characterisation of finite sized lattice boundary effects. A small number of other possible future applications and modifications were briefly described.

Chapter 5

Conclusion

This thesis has introduced algorithms for arbitrary graph modification that are generalisations of earlier methods. The new algorithms employ more complex data structures involving auxiliary objects, termed graph objects and vertex objects, as an efficiency device. These objects significantly reduce the computation required to determine whether or not a graph will been split in two if a specified edge is removed.

The algorithms were applied to the classical percolation model, where they generalise existing unidirectional microcanonical ensemble sampling methods to a bidirectional method that can operate indefinitely. This provides another efficiency gain in that the Markov chain Monte Carlo sampling random walk through configuration states can now be confined entirely within a small region of interest, instead of having to make a long approach to that region from an initially empty lattice.

The main objective was to make a high precision determination of the site percolation threshold for the two-dimensional square lattice. It has been shown how the approach adopted here has an advantage over some hull-walk methods in that it allows analysis of directional spanning property data without needing to choose a direction in advance, thereby providing a reduction in statistical uncertainty from an equivalent number of samples. It was also shown that data sampled from the microcanonical and canonical ensembles produce equivalent results as the lattice size tends to infinity, thereby eliminating the need to perform the Newman-Ziff convolution except when very high precision is required.

With the new algorithms and state-of-the-art pseudorandom number

generators, very high precision was indeed achieved, and the first ever measurement of the percolation threshold to seven figures was obtained. The result was of significant interest to the percolation community since the value was lower than expected. The result however, was reproducible, and has since been independently confirmed.

The discrepancy between this new result and earlier estimates has been explained, with an application specific test also introduced in this work, as the manifestation of correlations in the output sequences of the pseudorandom number generators used in the earlier works. The same test found no evidence that the generators used in this study have any such shortcomings, and so a certain level of guarantee can be made as to the new results accuracy. This is also a first, as previous works have simply had to take faith in their chosen generators, based on general grounds.

This percolation exercise has produced two sole-authorship papers in *Physical Review*, both of which have published citations, are referenced by Wikipedia, and feature on the BlueFern website.

This thesis has also computed a phase diagram for the exciton transport problem (based on the site-bond percolation model), and made a short study of the severity of boundary effects upon percolation clusters.

Although any of these problems could have been addressed with earlier methods, rather than those introduced here, this is often less convenient, more cumbersome, or computationally slower. On the other hand, the new method suffers from large memory requirements (even with the memory saving device of graph objects), introduces correlation problems into the data analysis, and requires much more complex source code (with all the headaches that introduces) than is usual in the field. However, if one is prepared to accept and accommodate these shortcomings, then demonstrably there is merit to the new algorithms.

References

[1] E. Abrahams, P. W. Anderson, D. C. Licciardello, and T. V. Ramakrishnan. Scaling theory of localization: Absence of quantum diffusion in two dimensions. *Physical Review Letters*, 42, 673 (1979).

[2] A. Aharony and J.-P. Hovi. Comment on "Spanning probability in 2D percolation". *Physical Review Letters*, 72, 1941 (1994).

[3] S. Alexander and R. Orbach. Density of states on fractals: fractons. *Journal of Physics (Paris) Letters*, 43, L623 (1982).

[4] Z. Alexandrowicz. Critically branched chains and percolation clusters. *Physics Letters A*, 80, 284 (1980).

[5] P. W. Anderson. Absence of diffusion in certain random lattices. *Physical Review*, 109, 1492 (1958).

[6] L. de Arcangelis, S. Redner, and A. Coniglio. Anomalous voltage distribution of random resistor networks and a new model for the backbone at the percolation threshold. *Physical Review B*, 31, 4725 (1985).

[7] G. B. Arfken and H. J. Weber. *Mathematical Methods for Physicists*. 5e, Harcourt Academic Press (2001).

[8] P. Bak. *How nature works: The Science of Self-Organised Criticality*. Copernicus (1996).

[9] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of the 1/f noise. *Physical Review Letters*, 59, 381 (1987).

[10] P. Balister, B. Bollobás, and M. Walters. Continuum Percolation with Steps in the Square or the Disc. *Random Structures and Algorithms*, 26, 392 (2005).

[11] H. G. Ballesteros, L. A. Fernández, V. Martín-Mayor, A. Muñoz Sudupe, G. Parisi, and J. J. Ruiz-Lorenzo. Scaling corrections: site percolation and Ising model in three dimensions. *Journal of Physics A*, 32, 1 (1999).

[12] A. A. Barker. Monte Carlo calculations of the radial distribution functions for a proton-electron plasma. *Australian Journal of Physics*, 18, 119 (1965).

[13] J. van den Berg. Percolation theory on pairs of matching lattices. *Journal of Mathematical Physics*, 22, 152 (1981).

[14] D. J. Bergman. Self-Duality and the Low Field Hall Effect in 2D and 3D Metal-Insulator Composites. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[15] L. Berlyand and J. Wehr. The probability distribution of the percolation threshold in a large system. *Journal of Physics A*, 28, 7127 (1995).

[16] P. R. Berman. Validity condtions for the optical bloch equations. *Journal of the Optical Society of America B*, 3, 564 (1986).

[17] K. Binder. *Applications of the Monte Carlo Method in Statistical Physics*. 2e, Springer (1987).

[18] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman. *The Theory of Critical Phenomena: an introduction to the renormalization group*, Oxford University Press (1992).

[19] R. Blanc and E. Guyon. Transport Properties of an Assembly of Spheres. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[20] F. Bloch. Nuclear induction. *Physical Review*, 70, 460 (1946).

[21] H. W. J. Blöte. Personal communication.

[22] B. Bollobás and O. Riordan. *Percolation*. Cambridge University Press (2006).

[23] W. W. Brandt. Use of percolation theory to estimate effective diffusion coefficients of particles migrating on various ordered lattices and in a random network structure. *The Journal of Chemical Physics*, 63, 5162 (1975).

[24] R. P. Brent. Some long-period random number generators using shifts and xors. *Australian and New Zealand Industrial and Applied Mathematics Journal*, 48, C188 (2007).

[25] S. R. Broadbent and J. M. Hammersley. Percolation processes. I. Crystals and mazes. *Proceedings of the Cambridge Philosophical Society*, 53, 629 (1957).

[26] T. W. Burkhardt and B. W. Southern. Renormalisation-group results for bond and site percolation in two and three dimensions. *Journal of Physics A*, 11, L253 (1978).

[27] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: percolation on random graphs. *Physical Review Letters*, 85, 5468 (2000).

[28] J. L. Cardy. Critical percolation in finite geometries. *Journal of Physics A*, 25, L201 (1992).

[29] J. L. Cardy and R. M. Ziff. Exact Results for the Universal Area Distribution of Clusters in Percolation, Ising, and Potts Models. *Journal of Statistical Physics*, 110, 1 (2003).

[30] D. M. Ceperley and B. J. Alder. Ground state of the electron gas by a stochastic method. *Physical Review Letters*, 45, 566 (1980).

[31] G. J. Chaitin. *Information, Randomness and Incompleteness*. World Scientific (1990).

[32] R. Chandler, J. Koplik, K. Lerman, and J. F. Willemsen. Capillary displacement and percolation in porous media. *Journal of Fluid Mechanics*, 119, 249 (1982).

[33] K. C. Chang and T. Odagaki. Site-Bond Percolation Problems. *Journal of Statistical Physics*, 35, 507 (1983).

[34] H Chaté, F. Ginelli, G. Grégoire, F. Peruani, and F. Raynaud. Modeling collective motion: variations on the Vicsek model. *The European Physical Journal B*, 64, 451 (2008).

[35] J. T. Chayes and L. Chayes. Percolation and random media. In K. Osterwalder and R. Stora, editors, *Critical Phenomena, Random Systems, Gauge Theories*. North-Holland (1986).

[36] J. T. Chayes, A. Puha, and T. Sweet. Independent and dependent percolation. In E. P. Hsu and S. R. S. Varadhan, editors, *Probability Theory and Applications*. American Mathematical Society (1999).

[37] W. G. Chase. See reference [277].

[38] W. K. Ching and M. K. Ng. *Markov Chains: Models, Algorithms and Applications*. Springer (2006).

[39] A. Compagner. Operational conditions for random-number generation. *Phyiscal Review E*, 52, 5634 (1995).

[40] A. Compagner and A. Hoogland. Maximum-length sequences, cellular automata, and random numbers. *Journal of Computational Physics*, 71, 391 (1987).

[41] A. Coniglio. Cluster structure near the percolation threshold. *Journal of Physics A*, 15, 3829 (1982).

[42] A. Coniglio and J. W. Essam. Percolation theory in a gas. *Journal of Physics A*, 10, 1917 (1977).

[43] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill (1990).

[44] M. Creutz. Microcanonical Monte Carlo Simulation. *Physical Review Letters*, 50, 1411 (1983).

[45] P. Dean. A new Monte Carlo method for percolation problems on a lattice. *Proceedings of the Cambridge Philosophical Society*, 59, 397 (1963).

[46] Y. Deng and H. W. J. Blöte. Monte Carlo study of the site-percolation model in two and three dimensions. *Physical Review E*, 72, 16126 (2005).

[47] Y. Deng, H. W. J. Blöte, and B. Nienhuis. Backbone exponents of the two-dimensional q-state Potts model: A Monte Carlo investigation. *Physical Review E*, 69, 26114 (2004).

[48] B. Derrida and L. de Seze. Application of the phenomenological renormalization to percolation and lattice animals in dimension 2. *Journal de Physique*, 43, 475 (1982).

[49] B. Derrida and D. Stauffer. Corrections to scaling and phenomenological renormalization for 2-dimensional percolation and lattice animal problems. *Journal de Physique*, 46, 1623 (1985).

[50] G. Deutscher, A. Kapitulnik, and M. Rappaport. Percolation in Metal-Insulator Systems. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[51] G. Deutscher, R. Zallen, and J. Adler. *Percolation Structures and Processes*. Israel Physical Society (1983).

[52] R. G. DeVoe and R. G. Brewer. Experimental test of the optical bloch equations for solids. *Physical Review Letters*, 50, 1269 (1983).

[53] Z. V. Djordjevic, H. E. Stanley, and A. Margolina. Site percolation threshold for honeycomb and square lattices. *Journal of Physics A*, 15, L405 (1982).

[54] C. Domb. The Percolation Phase Transition. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[55] C. Domb and M. F. Sykes. Cluster Size in Random Mixtures and Percolation Processes. *Physical Review*, 122, 77 (1961).

[56] Y. Dubi, Y. Meir, and Y. Avishai. Quantum hall criticality, superconductor-insulator transition, and quantum percolation. *Physical Review B*, 71, 125311 (2005).

[57] B. Duplantier. Conformally invariant fractals and potential theory. *Physical Review Letters*, 84, 1363 (2000).

[58] B. Duplantier. Conformal fractal geometry and boundary quantum gravity. *Proceedings of Symposia in Pure Mathematics*, 72, 365 (2004).

[59] R. J. Elliott, B. R. Heap, D. J. Morgan, and G. S. Rushbrooke. Equivalence of the Critical Concentrations in the Ising and Heisenberg Models of Ferromagnetism. *Physical Review Letters*, 5, 366 (1960).

[60] J. W. Essam. Percolation Processes. II. The Pair Connectedness. *Journal of Mathematical Physics*, 12, 874 (1971).

[61] J. W. Essam. Percolation and Cluster Size. In C. Domb and J. L. Lebowitz, editors, *Phase Transitions and Critical Phenomena*. Volume 2, Academic Press (1972).

[62] J. W. Essam. Percolation theory. *Reports on Progess in Physics*, 43, 833 (1980).

[63] S. N. Evangelou. Quantum percolation and the anderson transition in dilute systems. *Physical Review B*, 27, 1397 (1983).

[64] A. Fairall, D. Turner, M. L. Pretorius, M. Wiehahn, V. McBride, G. de Vaux, and P. A. Woudt. Percolation Properties of Nearby Large-Scale Structures: Every Galaxy has a Neighbour. *Arxiv eprint*, `astro-ph/0411437` (2004).

[65] A. P. Fairall and P. A. Woudt. *Nearby Large Scale Structures and the Zone of Avoidance*. Astronomical Society of the Pacific (2005).

[66] X. Feng, Y. Deng, and H. W. J. Blöte. Percolation transitions in two dimensions. *Physical Review E*, 78, 031136 (2008).

[67] A. M. Ferrenberg, and D. P. Landau, and Y. J. Wong. Monte Carlo simulations: Hidden errors from "good" random number generators. *Physical Review Letters*, 69, 3382 (1992).

[68] M. E. Fisher. The theory of equilibrium critical phenomena. *Reports on Progress in Physics*, 30, 615 (1967).

[69] M. E. Fisher and J. W. Essam. Some Cluster Size and Percolation Problems. *Journal of Mathematical Physics*, 2, 609 (1961).

[70] P. J. Flory. Molecular Size Distribution in Three Dimensional Polymers. I. Gelation. *Journal of the American Chemical Society*, 63, 3083 (1941).

[71] Th. Förster. Zwischenmolekulare energiewanderung und fluoreszenz. *Annalen der Physik*, 6, 55 (1948).

[72] J. E. de Freitas and L. D. S. Lucena. Equivalence between the FLR Time Dependent Percolation Model and the Newman-Ziff Algorithm. *International Journal of Modern Physics C*, 11, 1581 (2000).

[73] J. E. de Freitas, L. D. S. Lucena, and S. Roux. Percolation as a Dynamical Phenomenon. *Physica A*, 266, 81 (1999).

[74] J. Frenkel. On the transformation of light into heat in solids. I. *Physical Review*, 37, 17 (1931).

[75] H. L. Frisch and J. M. Hammersley. Percolation Processes and Related Topics. *Journal of the Society for Industrial and Applied Mathematics*, 11, 894 (1963).

[76] H. L. Frisch, E. Sonnenblick, V. A. Vyssotsky, and J. M. Hammersley. Critical Percolation Probabilities (Site Problem). *Physical Review*, 124, 1021 (1961).

[77] J. Fröhlich and T. Spencer. Absence of diffusion in the anderson tight binding model for large disorder or low energy. *Communications in Mathematical Physics*, 88, 151 (1983).

[78] G. Grimmett. The random-cluster model. In H. Kesten, editor, *Probability on Discrete Structures*. Springer (2004).

[79] B. M. Gammel. Hurst's rescaled range statistical analysis for pseudorandom number generators used in physical simulations. *Physical Review E*, 58, 2586 (1998).

[80] A. Gamsa and J. L. Cardy. SLE in the three-state Potts model-a numerical study. *Arxiv eprint*, `cond-mat/0705.1510` (2007).

[81] C. W. Gardiner. *Handbook of Stochastic Methods: for physics, chemistry and the natural sciences*. Springer-Verlag (1983).

[82] T. Gebele. Site percolation threshold for square lattice. *Journal of Physics A*, 17, L51 (1984).

[83] P. G. de Gennes. La percolation: un concept unificateur. *La Recherche*, 7, 919 (1976).

[84] M. R. Giri, M. J. Stephen, and G. S. Grest. Spin models and cluster distributions for bond and site percolation models. *Physical Review B*, 16, 4971 (1977).

[85] J. Goldenberg, B. Libai, S. Solomon, N. Jan, and D. Stauffer. Marketing percolation. *Physica A*, 284, 335 (2000).

[86] C. W. Goose. See reference [37].

[87] H. Gould and J. Tobochnik. *An Introduction to Computer Simulation Methods: Applications to Physical Systems*. 2e, Addison-Wesley (1996).

[88] H. Gould, J. Tobochnik, and W. Christian. *An Introduction to Computer Simulation Methods: Applications to Physical Systems*. 3e, Addison-Wesley (2006).

[89] P. Grassberger. Critical behavior of the general epidemic process and dynamical percolation. *Mathematical Biosciences*, 63, 157 (1983).

[90] P. Grassberger. On the hull of two-dimensional percolation clusters. *Journal of Physics A*, 19, 2675 (1986).

[91] P. Grassberger. Spreading and backbone dimensions of 2 D percolation. *Journal of Physics A*, 25, 5475 (1992).

[92] P. Grassberger. Conductivity exponent and backbone dimension in 2-d percolation. *Physica A*, 262, 251 (1999).

[93] P. Grassberger and Y. C. Zhang. Self-organized formulation of standard percolation phenomena. *Physica A*, 224, 169 (1996).

[94] J. Gravner. Growth Phenomena in Cellular Automata. In D. Griffeath and C. Moore, editors, *New Constructions in Cellular Automata*. Oxford University Press (2003).

[95] J. Gravner and D. Griffeath. First Passage Times for Threshold Dynamics on Z^2. *The Annals of Probability*, 24, 1752 (1996).

[96] J. Gravner and A. E. Holroyd. Slow convergence in bootstrap percolation. *Annals of Applied Probability*, 18, 909 (2008).

[97] G. S. Grest and M. H. Cohen. Percolation and the Glass Transition. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[98] D. J. Griffiths. *Introduction to Quantum Mechanics*. Prentice-Hall (1995).

[99] G. Grimmett. *Percolation*. Springer-Verlag (1989).

[100] T. Grossman and A. Aharony. Structure and perimeters of percolation clusters. *Journal of Physics A*, 19, L745 (1986).

[101] I. A. Gruzberg. Stochastic geometry of critical curves, Schramm–Loewner evolutions and conformal field theory. *Journal of Physics A*, 39, 12601 (2006).

[102] A. Haji-Akbari and R. M. Ziff. Percolation in networks with voids and bottlenecks. *Physical Review E*, 79, 021118 (2009).

[103] H. Haken and S. Nikitine. *Excitons at High Density*. Springer-Verlag (1975).

[104] J. M. Hammersley. Origins of Percolation Theory. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[105] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Methuen (1964).

[106] A. B. Harris, T. C. Lubensky, W. K. Holcomb, and C. Dasgupta. Renormalization-Group Approach to Percolation Problems. *Physical Review Letters*, 35, 327 (1975).

[107] T. E. Harris. A lower bound for the critical probability in a certain percolation process. *Proceedings of the Cambridge Philosophical Society*, 56, 13 (1960).

[108] P. Harrison. *Computational Methods is Physics, Chemsitry and Biology: An Introduction*. Wiley (2001).

[109] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97 (1970).

[110] S. Havlin and R. Nossal. Topological properties of percolation clusters. *Journal of Physics A*, 17, L427 (1984).

[111] C. L. Henley. Statics of a self-organizedpercolation model. *Physical Review Letters*, 71, 2741 (1993).

[112] H. J. Herrmann, B. Derrida, and J. Vannimenus. Superconductivity exponents in two-and three-dimensional percolation. *Physical Review B*, 30, 4080 (1984).

[113] H. J. Herrmann, D. C. Hong, and H. E. Stanley. Backbone and elastic backbone of percolation clusters obtained by the new method of 'burning'. *Journal of Physics A*, 17, L261 (1974).

[114] D. C. Hong, S. Havlin, H. J. Herrmann, and H. E. Stanley. Breakdown of Alexander-Orbach conjecture for percolation: Exact enumeration of random walks on percolation backbones. *Physical Review B*, 30, 4083 (1984).

[115] J. Hoshen and R. Kopelman. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. *Physical Review B*, 14, 3438 (1976).

[116] J.-P. Hovi and A. Aharony. Scaling and universality in the spanning probability for percolation. *Physical Review E*, 53, 235 (1996).

[117] C. K. Hu. Personal communication.

[118] C.-K. Hu. Histogram Monte Carlo renormalization group method for phase transition models without critical slowing down. *Physical Review Letters*, 69, 2739 (1992).

[119] C.-K. Hu. Histogram Monte Carlo Approach to Scaling Functions and Order Parameters of PHase Transition Models. *Chinese Journal of Physics*, 32, 519 (1994).

[120] C.-K. Hu. Large-cell renormalization group and order parameter for site percolation problems. *Physical Review B*, 51, 3922 (1995).

[121] C.-K. Hu, C.-N. Chen, and F. Y. Wu. Histogram Monte Carlo Poisiton-Space Renormalization Group: Applications to the Site Percolation. *Journal of Statistical Physics*, 82, 1199 (1996).

[122] J. Huang, J. M. Zhang, A. Lezama, and T. W. Mossberg. Excess dephasing in photon-echo experiments arising from excitation-induced electronic level shifts. *Physical Review Letters*, 63, 78 (1989).

[123] N. Jan and D. Stauffer. Random Site Percolation in Three Dimensions. *International Journal of Modern Physics C*, 9, 341 (1998).

[124] B. Jouhier, C. Allain, B. Gauthier-Manuel, and E. Guyon. The Sol-Gel Transition. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[125] B. Jovanović S. V. Buldyrev, S. Havlin, and H. E. Stanley. Punctuated equilibrium and "history-dependent" percolation. *Physical Review E*, 50, 2403 (1994).

[126] A. Kapitulnik, A. Aharony, G. Deutscher, and D. Stauffer. Self similarity and correlations in percolation. *Journal of Physics A*, 16, L269 (1983).

[127] R. F. Kazarinov and S. Luryi. Quantum percolation and quantization of hall resistance in two-dimensional electron gas. *Physical Review B*, 25, 7626 (1982).

[128] V. M. Kenkre and P. Reineker. *Exciton Dynamics in Molecular Crystals and Aggregates*. Springer-Verlag (1982).

[129] A. E. Kennelly. *Artificial Electric Lines: Their Theory, Mode of Construction and Uses*. McGraw-Hill (1917).

[130] J. Kertész. Extrapolation of transfer matrix data for percolation and lattice animals by the Romberg-Beleznay algorithm. *Journal of Physics A*, 19, 599 (1986).

[131] H. Kesten. The critical probability of bond percolation on the square lattice equals 1/2. *Communications in Mathematical Physics*, 74, 41 (1980).

[132] H. Kesten. *Percolation Theory for Mathematicians*. Birkhäuser (1982).

[133] H. Kesten. First-passage percolation. In P. Picco and J. San Martin, editors, *From Classical to Modern Probability*. Birkhäuser (2003).

[134] W. Kinzel. Directed Percolation. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[135] S. Kirkpatrick and T. P. Eggarter. Localized states of a binary alloy. *Physical Review B*, 6, 3598 (1972).

[136] S. Kirkpatrick and E. P. Stoll. A Very Fast Shift-Register Sequence Random Number Generator. *Journal of Computational Physics*, 40, 517 (1981).

[137] P. Kleban, J. J. H. Simmons, and R. M. Ziff. Anchored Critical Percolation Clusters and 2D Electrostatics. *Physical Review Letters*, 97, 115702 (2006).

[138] P. Kleban and R. M. Ziff. Exact results at the two-dimensional percolation point. *Physical Review B*, 57, 8075 (1998).

[139] W. Klein, H. E. Stanley, P. J. Reynolds, and A. Coniglio. Renormalization-Group Approach to the Percolation Properties of the Triangular Ising Model. *Physical Review Letters*, 41, 1145 (1978).

[140] D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. 2e, Addison-Wesley (1981).

[141] A. N. Kolmogorov. Combinatorial foundations of information theory and the calculus of probabilities. *Russian Mathematical Surveys*, 38, 29 (1983).

[142] Th. Koslowski and W. von Niessen. Mobility edges for the quantum percolation problem in two and three dimensions. *Physical Review B*, 42, 10342 (1990).

[143] M. Kurant and P. Thiran. Error and Attack Tolerance of Layered Complex Networks. *Arxiv eprint*, `physics/0610018` (2006).

[144] R. G. Larson, L. E. Scriven, and H. T. Davis. Percolation theory of residual phases in porous media. *Nature*, 268, 409 (1977).

[145] B. J. Last and D. J. Thouless. Percolation Theory and Electrical Conductivity. *Physical Review Letters*, 27, 1719 (1971).

[146] P. L. Leath. Cluster size and boundary distribution near percolation threshold. *Physical Review B*, 14, 5046 (1976).

[147] P. L'Ecuyer. *Random Number Generation*. Wiley (1997).

[148] P. L'Ecuyer. Tables of linear congruential generators of different sizes and good lattice structure. *Mathematics of Computation*, 68, 249 (1999).

[149] P. L'Ecuyer and R. Simard. TestUO1: A C Library for Empirical Testing of Random Number Generators. *American Mathematical Society Transactions on Mathematical Software*, 33, 22 (2007).

[150] D.-H. Lee, Z. Wang, and S. Kivelson. Quantum percolation and plateau transitions in the quantum hall effect. *Physical Review Letters*, 70, 4130 (1993).

[151] M. J. Lee, M. F. Reid, M. D. Faucher, and G. W. Burdick. Comparison between correlation crystal field calculations using extended basis sets and two-electron operators. *Journal of Alloys and Compounds*, 323-324, 636 (2001).

[152] M. J. Lee. Complementary algorithms for graphs and percolation. *Physical Review E*, 76, 027702 (2007).

[153] M. J. Lee. Pseudo-random-number generators and the square site percolation threshold. *Physical Review E*, 78, 031131 (2008).

[154] D.H. Lehmer. Mathematical methods in large-scale computing units. In H. D. Huskey, editor, *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery*. Harvard University Press (1951).

[155] R. Lenormand and S. Bories. Description d'un mécanisme de connexion de liaison destiné à l'étude du drainage avec piégeage en milieu poreux. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences de Paris, Série B*, 291, 279 (1980).

[156] T. G. Lewis and W. H. Payne. Generalized Feedback Shift Register Pseudorandom Number Algorithm. *Journal of the Association for Computing Machinery*, 20, 456 (1973).

[157] C. Y. Lin and C. K. Hu. Universal finite-size scaling functions for percolation on three-dimensional lattices. *Physical Review E*, 58, 1521 (1998).

[158] G. K. Liu, M. F. Joubert, R. L. Cone and B. Jacquier. Photon echo relaxation and hole burning in Tb^{3+}:$LiYF_4$ and $LiTbF_4$. *Journal of Luminescence*, 38, 34 (1987).

[159] C. J. Lobb and D. J. Frank. Percolative conduction and the Alexander-Orbach conjecture in two dimensions. *Physical Review B*, 30, 4090 (1984).

[160] C. D. Lorenz and R. M. Ziff. Precise determination of the bond percolation thresholds and finite-size scaling corrections for the sc, fcc, and bcc lattices. *Physical Review E*, 57, 230 (1998).

[161] R. F. Loring, H. C. Anderson, and M. D. Fayer. Theory of photon echoes from interacting impurities in crystals with inhomogeneously broadened absorption spectra. *Journal of Chemical Physics*, 81, 5395 (1984).

[162] R. M. Macfarlane and R. M. Shelby. *Spectroscopy of Solids containing Rare Earth Ions*. North-Holland (1987).

[163] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. M. Chayes. Invaded cluster algorithm for potts models. *Physical Review E*, 54, 1332 (1996).

[164] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. V. Chayes. Invaded cluster algorithm for equilibrium critical points. *Physical Review Letters*, 75, 2792 (1995).

[165] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman (1983).

[166] B. B. Mandelbrot. An Explicit Fractal Model of Percolation Clusters. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[167] A. Margolina, Z. V. Djordjevic, D. Stauffer, and H. E. Stanley. Corrections to scaling for branched polymers and gels. *Physical Review B*, 28, 1652 (1983).

[168] G. Marsaglia. The structure of linear congruential sequences. In S. K. Zaremba, editor, *Applications of Number Theory to Numerical Analysis*. Academic Press (1972).

[169] G. Marsaglia. A current view of random number generators. In L. Billard, editor, *Computer Science and Statistics: the Interface*. Elsevier (1985).

[170] G. Marsaglia. Random numbers for C: End, at last? `sci.stat.math` web discussion (1999).

[171] G. Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8, 1 (2003).

[172] G. Marsaglia and A. Zaman. A New Class of Random Number Generators. *The Annals of Applied Probability*, 1, 462 (1991).

[173] J. Martín-Herrero. Hybrid cluster identification. *Journal of Physics A*, 37, 9377 (2004).

[174] P. H. L. Martins. Personal communication.

[175] P. H. L. Martins and J. A. Plascak. Percolation on two- and three- dimensional lattices. *Physical Review E*, 67, 046119 (2003).

[176] M. Matsumoto and Y. Kurita. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *Association for Computing Machinery Transactions on Modeling and Computer Simulation*, 6, 99 (1996).

[177] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *Association for Computing Machinery Transactions on Modeling and Computer Simulation*, 8, 3 (1998).

[178] R. Meester and R. Roy. *Continuum Percolation*. Cambridge University Press (1996).

[179] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21, 1087 (1953).

[180] H. Mino, M. Yamamoto, T. Kawai, I. Akai, and T. Karasawa. Anomalous behaviour of the high-density excitons and their nonlinear optical response in a quasi-two-dimensional space in a layered crystal bii$_3$. *Journal of Luminescence*, 87-89, 278 (2000).

[181] C. D. Mitescu. Diffusion on Percolation Clusters. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[182] M. Mitsunaga, T. Takagahara, R. Yano, and N. Uesugi. Excitation-Induced Frequency Shift Probed by Stimulated Photon Echoes. *Physical Review Letters*, 68, 3216 (1992).

[183] A. Mookerjee, I. Dasgupta, and T. Saha. Quantum Percolation. *International Journal of Modern Physics B*, 9, 2989 (1995).

[184] C. Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61, 5678 (2000).

[185] C. Moukarzel. A Fast Algorithm for Backbones. *International Journal of Modern Physics C*, 9, 887 (1998).

[186] H. Nakanishi and P. J. Reynolds. Site-bond percolation by position-space renormalization group. *Physics Letters A*, 71, 252 (1979).

[187] H. Nakanishi and H. E. Stanley. Scaling studies of percolation phenomena in systems of dimensionality two to seven: Cluster numbers. *Physical Review B*, 22, 2466 (1980).

[188] J. von Neumann and A. W. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press (1966).

[189] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64, 026118 (2001).

[190] M. E. J. Newman and R. M. Ziff. Efficient Monte Carlo Algorithm and High-Precision Results for Percolation. *Physical Review Letters*, 85, 4104 (2000).

[191] M. E. J. Newman and R. M. Ziff. Fast Monte Carlo algorithm for site or bond percolation. *Physical Review E*, 64, 016706 (2001).

[192] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics (1992).

[193] I. M. Nolden and R. J. Silbey. Simulation of spin diffusion in a disordered system. *Physical Review B*, 54, 381 (1996).

[194] T. Odagaki, N. Ogita, and H. Matsuda. Quantal percolation problems. *Journal of Physics C*, 13, 189 (1980).

[195] P. M. C. de Oliveira. Personal communication.

[196] P. M. C. de Oliveira, R. A. Nóbrega, and D. Stauffer. Corrections to Finite Size Scaling in Percolation. *Brazillian Journal of Physics*, 33, 616 (2003).

[197] F. Panneton, P. L'Ecuyer, and M. Matsumoto. Improved long-period generators based on linear recurrences modulo 2. *Association for Computing Machinery Transactions on Mathematical Software*, 32, 1 (2006).

[198] J. Park and M. E. J. Newman. Statistical mechanics of networks. *Physical Review E*, 70, 066117 (2004).

[199] R. Parviainen. Estimation of Bond Percolation Thresholds on the Archimedean Lattices. *Arxiv eprint*, `cond-mat/0704.2098` (2007).

[200] G. Paul, R. M. Ziff, and H. E. Stanley. Percolation threshold, Fisher exponent, and shortest path exponent for four and five dimensions. *Physical Review E*, 64, 26115 (2001).

[201] E. Perlsman and S. Havlin. Method to estimate critical exponents using numerical studies. *Europhysics Letters*, 58, 176 (2002).

[202] R. Pike and H. E. Stanley. Order propagation near the percolation threshold. *Journal of Physics A*, 14, L169 (1981).

[203] H. T. Pinson. Critical percolation on the torus. *Journal of Statistical Physics*, 75, 1167 (1994).

[204] A. M. Portis. Spectral diffusion in magnetic resonance. *Physical Review*, 104, 584 (1956).

[205] A. Proykova and D. Stauffer. Social percolation and the influence of mass media. *Physica A*, 312, 300 (2002).

[206] S. Quintanilla, S. Torquato, and R. M. Ziff. Efficient measurement of the percolation threshold for fully penetrable discs. *Journal of Physics A*, 33, L399 (2000).

[207] R. Rammal, J. C. Angles d'Auriac, and A. Benoit. Universality of the spectral dimension of percolation clusters. *Physical Review B*, 30, 4087 (1984).

[208] D. C. Rapaport. Monte Carlo experiments on percolation: the influence of boundary conditions. *Journal of Physics A*, 18, L175 (1985).

[209] S. Redner. Percolation and Conduction in Random Resistor-Diode Networks. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[210] P. J. Reynolds, H. E. Stanley, and W. Klein. A real-space renormalization group for site and bond percolation. *Journal of Physics C*, 10, L167 (1977).

[211] P. J. Reynolds, H. E. Stanley, and W. Klein. Percolation by position-space renormalisation group with large cells. *Journal of Physics A*, 11, L199 (1978).

[212] P. J. Reynolds, H. E. Stanley, and W. Klein. Large-cell monte carlo renormalization group for percolation. *Physical Review B*, 21, 1223 (1980).

[213] M. D. Rintoul and H. Nakanishi. A precise determination of the backbone fractal dimension on two-dimensional percolation clusters. *Journal of Physics A*, 25, L945 (1992).

[214] O. Riordan and M. Walters. Rigorous confidence intervals fopr critical probabilities. *Physical Review E*, 76, 011110 (2007).

[215] L. J. Root and J. L. Skinner. Optical dephasing and photon echoes from energetically and substitutionally disordered crystals. *Journal of Chemical Physics*, 81, 5310 (1984).

[216] L. J. Root and J. L. Skinner. Frequency-dependent optical dephasing and the nature of inhomogeneous broadening in crystals. *Physical Review B*, 32, 4111 (1985).

[217] L. J. Root and J. L. Skinner. Localization phase diagram for the energetically and substitutionally disordered anderson/quantum percolation model. *Journal of Chemical Physics*, 89, 3279 (1988).

[218] M. Rosso, J. F. Gouyet, and B. Sapoval. Determination of percolation probability from the use of a concentration gradient. *Physical Review B*, 32, 6053 (1985).

[219] J. Roussenq, J. Clerc, G. Girand, E. Guyon, and H. Ottavi. Size Dependence of Percolation Threshold of Square and Triangular Networks. *Journal de Physique Letters*, 37, (1976).

[220] L. Russo. A note on percolation. *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete*, 43, 39 (1978).

[221] M. Sahimi and J. D. Goddard. Elastic percolation models for cohesive mechanical failure in heterogeneous systems. *Physical Review B*, 33, 7848 (1986).

[222] M. Sahimi. *Applications of Percolation Theory*. Taylor & Francis (1994).

[223] J. J. Sakurai. *Modern Quantum Mechanics*. Addison-Wesley. (1994).

[224] H. Saleur and B. Duplantier. Exact Determination of the Percolation Hull Exponent in Two Dimensions. *Physical Review Letters*, 58, 2325 (1987).

[225] B. Sapoval, M. Rosso, and J. F. Gouyet. The fractal nature of a diffusion front and the relation to percolation. *Journal de Physique Lettres*, 46, 149 (1985).

[226] J. L. Schiff. *Cellular Automata: A Discrete View of the World*. Wiley-Interscience. (2008).

[227] O. Schramm. Scaling limits of loop-erased random walks and uniform spanning trees. *Israel Journal Mathematics*, 118, 221 (2001).

[228] L. S. Schulman and P. E. Seiden. Propagating Stochastic Star Formation and Galactic Structure. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[229] C. R. Scullard and R. M. Ziff. Predictions of bond percolation thresholds for the Kagomé and Archimedean (3, 12^2) lattices. *Physical Review E*, 73, 45102 (2006).

[230] P. E. Seiden and L. S. Schulman. Percolation model of galactic structure. *Advances in Physics*, 39, 1 (1990).

[231] P. D. Seymour and D. J. A. Welsh. Percolation probabilities on the square lattice. *Annals of Discrete Mathematics*, 3, 227 (1978).

[232] R. N. Shakhmuratov, and F. M. Gelardi, and M. Cannas. Non-Bloch Transients in Solids: Free Induction Decay and Transient Nutations. *Physical Review Letters*, 79, 2963 (1997).

[233] Y. Shapir, A. Aharony, and A. Brooks Harris. Localization and quantum percolation. *Physical Review Letters*, 49, 486 (1982).

[234] B. Shapiro. Real-space renormalisation in the percolation problem. *Journal of Physics C*, 12, 3185 (1979).

[235] B. Shapiro. Localization versus Percolation. G. Deutscher, R. Zallen, and J. Adlere, *Percolation Structures and Processes*. Israel Physical Society (1983).

[236] L. N. Shchur. Incipient Spanning Clusters in Square and Cubic Percolation. *Arxiv eprint*, `cond-mat/9906013` (1999).

[237] J. J. H. Simmons, P. Kleban, and R. M. Ziff. Percolation Crossing Formulae and Conformal Field Theory. *Journal of Physics A*, 40, F771 (2007).

[238] J. L. Skinner and D. Hsu. Optical dephasing of ions and molecules in crystals. *Advances in Chemical Physics*, 65, 1 (1986).

[239] S. Smirnov. Critical percolation in the plane. I. Conformal Invariance and Cardy's formula II. Continuum scaling limit. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série A*, 333, 239 (2001).

[240] S. Smirnov and W. Werner. Critical exponents for two-dimensional percolation. *Mathematical Research Letters*, 8, 729 (2001).

[241] S. Solomon, G. Weisbuch, L. de Arcangelis, N. Jan, and D. Stauffer. Social percolation models. *Physica A*, 277, 239 (2000).

[242] H. E. Stanley and A. Coniglio. Fractal Structure of the Incipient Infinite Cluster in Percolation. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[243] D. Stauffer. Exact distribution of cluster size and perimeter for two-dimensional percolation. *Zeitschrift fur Physik B*, 25, 391 (1976).

[244] D. Stauffer. Percolation clusters as teaching aid for Monte Carlo simulation and critical exponents. *American Journal of Physics*, 45, 1001 (1977).

[245] D. Stauffer. Scaling Theory of Percolation Clusters. *Physics Reports*, 54, 1 (1979).

[246] D. Stauffer and A. Aharony. *Introduction to Percolation Theory*. Taylor & Francis (1994).

[247] J. P. Straley. Scaling Predictions for Physical Properties. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[248] D. W. Stroock. *An Introduction To Markov Processes*. Springer (2005).

[249] P. N. Suding and R. M. Ziff. Site percolation thresholds for Archimedean lattices. *Physical Review E*, 60, 275 (1999).

[250] R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58, 86 (1987).

[251] M. F. Sykes and J. W. Essam. Some Exact Critical Percolation Probabilities for Bond and Site Problems in Two Dimensions. *Physical Review Letters*, 10, 3 (1963).

[252] M. F. Sykes and J. W. Essam. Critical Percolation Probabilities by Series Methods. *Physical Review*, 133, 310 (1964).

[253] M. F. Sykes and J. W. Essam. Exact Critical Percolation Probabilities for Site and Bond Problems in Two Dimensions. *Journal of Mathematical Physics*, 5, 1117 (1964).

[254] M. F. Sykes, D. S. Gaunt, and M. Glen. Percolation processes in three dimensions. *Journal of Physics A*, 9, 1705 (1976).

[255] K. Sznajd-Weron and J. Sznajd. Opinion Evolution in a Closed Community. *International Journal of Modern Physics C*, 11, 1157 (2000).

[256] R. E. Tarjan. Depth-First Search and Linear Graph Algorithms. *Society for Industrial and Applied Mathematics Journal on Computing*, 1, 146 (1972).

[257] R. E. Tarjan. Efficiency of a Good But Not Linear Set Union Algorithm. *Journal of the Association for Computing Machinery*, 22, 215 (1975).

[258] R. C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19, 201 (1965).

[259] J. P. G. Taylor and A. MacKinnon. A study of the two-dimensional bond quantum percolation model. *Journal of Physics: Condensed Matter*, 1, 9963 (1989).

[260] H. N. V. Temperley and E. H. Lieb. Relations between the Percolation and Colouring Problem and other Graph-Theoretical Problems Associated with Regular Planar Lattices: Some Exact Results for the Percolation Problem. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 322, 251 (1971).

[261] S. Tezuka, P. L'Ecuyer, and R. Couture. On the lattice structure of the add-with-carry and subtract-with-borrow random number generators. *Association for Computing Machinery Transactions on Modeling and Computer Simulation*, 3, 315 (1993).

[262] D. Tiggemann. Simulation of Percolation on Massively-Parallel Computers. *International Journal of Modern Physics C*, 12, 871 (2001).

[263] Y. Tomita and Y. Okabe. Probability-changing cluster algorithm for potts model. *Physical Review Letters*, 86, 572 (2001).

[264] S. A. Trugman. Localization, percolation, and the quantum hall effect. *Physical Review B*, 27, 7539 (1983).

[265] C. Tsallis. Phase diagram of anisotropic planar Potts ferromagnets: a new conjecture. *Journal of Physics C*, 15, L757 (1982).

[266] W. W. Tsang, L. C. K. Hui, K. P. Chow, C. F. Chong, and C. W. Tso. Tuning the collision test for power. *Proceedings of the 27th Australasian Conference on Computer Science*, 26, 23 (2004).

[267] D. L. Turcotte. *Fractals and chaos in geology and geophysics*. Cambridge University Press (1992).

[268] C. A. Voigt and R. M. Ziff. Epidemic analysis of the second-order transition in the Ziff-Gulari-Barshad surface-reaction model. *Physical Review E*, 56, 6241 (1997).

[269] R. F. Voss. The fractal dimension of percolation cluster hulls. *Journal of Physics A*, 17, L373 (1984).

[270] V. A. Vyssotsky, S. B. Gordon, H. L. Frisch, and J. M. Hammersley. Critical Percolation Probabilities (Bond Problem). *Physical Review*, 123, 1566 (1961).

[271] B. P. Watson and P. L. Leath. Conductivity in the two-dimensional-site percolation problem. *Physical Review B*, 9, 4893 (1974).

[272] A. Weinrib and S. A. Trugman. A new kinetic walk and percolation perimeters. *Physical Review B*, 31, 2993 (1985).

[273] E. W. Weisstein. *CRC Concise Encyclopedia of Mathematics*. Chapman & Hall/CRC (1999).

[274] J. C. Wierman. Bond Percolation on Honeycomb and Triangular Lattices. *Advances in Applied Probability*, 13, 298 (1981).

[275] J. C. Wierman. Counterexamples in percolation: the site percolation critical probabilities p_H and p_T are unequal for a class of fully triangulated graphs. *Journal of Physics A*, 17, 637 (1984).

[276] J. C. Wierman. A bond percolation critical probability determination based on the star-triangle transformation. *Journal of Physics A*, 17, 1525 (1984).

[277] G. C. Wild. See reference [86].

[278] U. Wolff. Collective monte carlo updating for spin systems. *Physical Review Letters*, 62, 361 (1989).

[279] S. Wolfram. *A New Kind of Science*. Wolfram Media (2002).

[280] F. Y. Wu. Critical point of planar Potts models. *Journal of Physics C*, 12, L645 (1979).

[281] F. Y. Wu. The Potts model. *Reviews of Modern Physics*, 54, 235 (1982).

[282] F. Yonezawa, S. Sakamoto, and M. Hori. Percolation in two-dimensional lattices. I. A technique for the estimation of thresholds. *Physical Review B*, 40, 636 (1989).

[283] J. G. Zabolitzky. Monte Carlo evidence against the Alexander-Orbach conjecture for percolation conductivity. *Physical Review B*, 30, 4077 (1984).

[284] R. Zallen. Introduction to Percolation: a model for all seasons. In G. Deutscher, R. Zallen, and J. Adler, editors, *Percolation Structures and Processes*. Israel Physical Society (1983).

[285] R. M. Ziff. Personal communication.

[286] R. M. Ziff. Test of scaling exponents for percolation-cluster perimeters. *Physical Review Letters*, 56, 545 (1986).

[287] R. M. Ziff. Spanning probability in 2D percolation. *Physical Review Letters*, 69, 2670 (1992).

[288] R. M. Ziff. Reply to comment on Spanning Probability in 2D percolation. *Physical Review Letters*, 72, 1942 (1994).

[289] R. M. Ziff. Proof of crossing formula for 2D percolation. *Journal of Physics A*, 28, 6479 (1995).

[290] R. M. Ziff. Effective boundary extrapolation length to account for finite-size effects in the percolation crossing function. *Physical Review E*, 54, 2547 (1996).

[291] R. M. Ziff. Four-tap shift-register-sequence random-number generators. *Computers in Physics*, 12, 385 (1998).

[292] R. M. Ziff. Generalized cell–dual-cell transformation and exact thresholds for percolation. *Physical Review E*, 73, 16134 (2006).

[293] R. M. Ziff. Algorithms for Percolation. In *Encyclopedia of Percolation*. (To be published).

[294] R. M. Ziff, P. T. Cummings, and G. Stell. Generation of percolation cluster perimeters by a random walk. *Journal of Physics A*, 17, 3009 (1984).

[295] R. M. Ziff, C. D. Lorenz, and P. Kleban. Shape-Dependent Universality in Percolation. *Physica A*, 266, 17 (1999).

[296] R. M. Ziff and M. E. J. Newman. Convergence of threshold estimates for two-dimensional percolation. *Physical Review E*, 66, 16129 (2002).

[297] R. M. Ziff and B. Sapoval. The efficient determination of the percolation threshold by a frontier-generating walk in a gradient. *Journal of Physics A*, 19, L1169 (1986).

[298] R. M. Ziff and C. R. Scullard. Exact bond percolation thresholds in two dimensions. *Journal of Physics A*, 39, 15083 (2006).

[299] R. M. Ziff and G. Stell. *Report No. 88-4*. Footnote 26. Laboratory for Scientific Computing, University of Michigan (1988).

[300] R. M. Ziff and P. N. Suding. Determination of the bond percolation threshold for the Kagomé lattice. *Journal of Physics A*, 30, 5351 (1997).

[301] B. H. Zimm and W. H. Stockmayer. The Dimensions of Chain Molecules Containing Branches and Rings. *Journal of Chemical Physics*, 17, 1301 (1949).

10^{16} pseudorandom numbers later, I'm seeing a pattern.